Design, Modeling, and Simulation of a Hydrogen-Induced Air Liquefaction System Applicable to Space Access Vehicles

John E. Crowley

AE 8900 Special Problem Report
May 2, 2004

School of Aerospace Engineering
Space Systems Design Lab
Georgia Institute of Technology
Atlanta, GA 30332-0150

Advisor: Dr. John R. Olds

Table of Contents

List of Figures

List of Tables

List of Acronyms

ACES	Air Collection and Enrichment System
ALS	Air Liquefaction System
DLL	Dynamic Linked Library
FPI	Fixed Point Iteration
HEX	Heat Exchanger
I_{sp}	Specific Impule
LACE	Liquid Air Cycle Engine
LAIR	Liquid Air
LEA	Liquid Enriched Air
LH2	Liquid Hydrogen
MDO	Multidisciplinary Design Optimization
ME	Model Engineer
OBD	Optimizer Based Decomposition
OE	Object Engineer
RBCC	Rocket Based Combined Cycle
SLS	Sea Level Static
WS	Water Separator

Abstract

Air liquefaction has been advocated as a concept for reducing the gross mass of spaceplanes since the 1950s. However, little has come of this promising technology thus far. The purpose of air liquefaction is to use cryogenic propellants to supercool air after it enters the inlet of a moving vehicle. The resulting liquefied air can be used immediately or stored for use in later stages of flight. Although the concept has been revisited multiple times since its inception, various problems have surfaced to keep it from being a viable technology for incorporation into an access-to-space vehicle. Among the many design challenges, one of these problems is the formation of solid ice due to the freezing of water in the humid air. This ice fouls heat exchanger surfaces and results in decreased efficiency and eventually total shut-down of the engine. Therefore, a system to dehumidify the air before it is used as an oxidizer or stored is necessary.

To combat this problem, an Air Liquefaction System (ALS) is proposed that utilizes a system of two water separators, four heat exchangers, and an atomized glycol spray inserted into the air stream in order to remove water from the humid incoming air. In the scope of this project, a computer tool will be created that models the effectiveness of such a system given inlet conditions. Utilizing the temperature, pressure, mass flow rate, and relative humidity of the incoming air stream, the required mass flow rate of liquid hydrogen needed to cool the air to a liquid state will be calculated for an ALS of specified size. The heat exchangers will all be of a counter-flow design. This software is of interest to and supported by employees of NASA Langley, SAIC, and Modelogics Inc., and is also of interest to the U.S. Air Force.

The primary system of interest is the ALS system model specified by Larry Hunt. This system models the ALS at sea level static (SLS) conditions. Air enters the inlet at 6000 lbm/min, a temperature of 540 R, a relative humidity of 80%, and a pressure of 14.7 psia. An appropriate mass flow of liquid hydrogen at 40 R and 800 psia must be added to the system so that the air at the exit of the ALS is liquid at a temperature around 130 R, and a pressure around 10 psia. Furthermore, most of the water must be removed from the flow utilizing the two water separators and the atomized ethylene glycol spray. The ratio of mass flow of liquid air produced to the mass flow of the hydrogen added to the system should be somewhere between 4 and 5.

The ALS model was created so that it could execute within both the Model Engineer and ModelCenter design frameworks. To this end, the components of the ALS were coded in Visual Basic and C++. These components were validated against existing components and actual hardware data. While the performance characteristics of each component match known values for that component fairly well, there are sometimes large errors in the weights and sizing components.

The components were assembled into the total system model, and the efficacy of the two approaches was compared. Executing the ALS model from ModelCenter proved to be the superior approach for optimization. The optimization was conducted for the conditions specified above, using Optimizer-Based Decomposition. The objective function was to minimize the system weight as a function of heat exchanger effectiveness values and liquid hydrogen mass flow. The optimization completed successfully, with a run time of about 5.5 minutes. The resulting ALS weighs 2783 lbm and produces 5935 lbm/min of liquid air at 129 R and 10.8 psia. This ALS requires a mass flow of liquid hydrogen of 1419 lb/min, for a total system efficiency of 4.18 lb air/lb LH2.

Introduction

Air liquefaction has been advocated as a technology for reducing the gross mass of space planes since the 1950s. The purpose of air liquefaction is to use cryogenic propellants to supercool the air entering an engine as the aircraft moves through the atmosphere. The resulting liquefied air can be immediately used or stored for use in later stages of flight. Liquid air (or LAIR) has a couple of advantages over gaseous air. In order to use air in a combustion system, it must be pressurized to the high pressure levels required for combustion. This requires heavy compressors; in contrast, a liquid oxidizer can be pumped to high pressures with basic turbomachinery for a much lower weight penalty. Additionally, for combined cycle engines, liquid oxidizer must often be stored and carried throughout the early stages of flight where it acts only as dead weight. If liquid air can be stored and separated into liquid nitrogen and oxygen, a vehicle can collect its oxidizer as it flies through the atmosphere, again making huge system weight savings.

In its most basic form, an Air Liquefaction System (ALS) consists of a precooler and a condenser. The precooler is used to bring the incoming air close to its liquefaction point, while the condenser does the work of actually changing the air's state to its liquid form. However, there are numerous other ways to structure an ALS, consisting of any number of heat exchangers and other components.

One such component is a water separator. There are a number of design issues that have prevented this technology from being realized and utilized in current systems, but one of the more prevalent challenges is the fouling of heat exchanger surfaces due to the formation of water ice from humid incoming air. This ice fouls heat exchanger surfaces and results in decreased efficiency and can eventually result in engine failure. By including a water separation capability into an ALS, this problem is neatly averted.

A number of programs already exist within the private and public sectors for analyzing heat exchangers, water separators, and other engine components. In the 1990s, Optimal Corporation designed LACEX for NASA Glenn Research Center. Additionally, SAIC created the Vehicle Thermal Management Analysis Code (VITMAC) (Ref XX) to analyze the operation of an air-liquefaction system. These programs model air-liquefaction, but have no capability to simulate removing water from the incoming air, nor are they conducive to methods of Multidisciplinary Design Optimization. Such methods make the analysis of a design space much easier and more rapid, and can greatly aid engineers in choosing a direction for a specific design.

Therefore the desired approach for modeling the ALS is one that can be executed within a modeling environment that allows the user to assemble different components into an overall system model. This model can then be analyzed at a number of design points, and optimized for different objective functions. Both Georgia Tech's Space Systems Design Lab and NASA have experience using Phoenix Integration's ModelCenter software. It has the capability to "wrap" user codes, and then conduct optimization upon system variables. The US Air Force has also had some recent experience with the Model Engineer software from Modelogics, Inc. This thermally-centered software also embraces the concept of "wrapping" user codes and allowing the construction of large systems from basic components, although its optimization capabilities are not as strong.

This project utilizes these two codes to create an ALS model. These two models can be compared, and the ALS performance studied. The completed models output the mass flows, pressures, temperatures, and other system properties to the user. The model may also be optimized for system parameters such as gross weight.

1 State of the Art in Air Liquefaction

Because air liquefaction does go back so many years, there has been a lot of work done already in the field. Therefore it is helpful to understand where the technology has been proposed to be applied, and what kind of progress has been made. Different types of engine cycles have been proposed that are capable of using liquid air as a propellant. These include the well-known LACE and ACES concepts, as well as more exotic configurations like SuperLACE, RamLACE, or ScramLACE. Depending on the type of cycle being analyzed, different components will be necessary; however, all cycles will require multiple heat exchangers, and there are many types available for this application. Because many of these cycles fly at low altitudes, fouling from water freezing upon heat exchanger surfaces out of the humid air becomes a problem. Therefore an approach to removing water from the air stream will also be required.

1.1 Liquid Air Engine Cycles

Air liquefaction is of interest today due to the number of combined cycle engine vehicle concepts being proposed. Whereas typical space vehicles utilize rockets throughout all stages of flight, these combined cycles will use different propulsion solutions through different regimes of flight, such that an optimal propulsive device is used at most points of a spaceplane’s trajectory. Rocket Based Combined Cycle (RBCC) engines generally use a rocket through the early and late stages of a vehicle’s flight. During supersonic flight, the vehicle switches to ramjet and scramjet (supersonic combustion ramjet) modes so that the oxygen in the atmosphere can be used for propulsion rather than carrying heavy oxygen in tanks for the rocket to use. There may be slight modifications to this basic design; the term “combined cycle” can apply to vehicles with two separate flowpaths, or vehicles with a single flow path. However, the idea remains the same. The cycles that employ air liquefaction are the basic LACE, ACES, and other exotic variations on the LACE concept.

1.1.1 Basic LACE

A Liquid Air Cycle Engine (LACE), pictured in Figure 1, is one of the simplest engine cycles proposed. Air enters an inlet, where it is decelerated and gains static pressure. It then enters the air liquefaction heat exchangers. There it is exposed to liquid hydrogen which cools the air until it reaches its liquid state and can be pumped to the combustion chamber. This hydrogen has already been pumped to the high pressures needed for combustion. By exchanging heat with the air, the liquid hydrogen becomes a high pressure gas. Once the air is liquefied and pumped, there are now two high pressure reactants ready to combust in the thrust chamber. This operation is very similar to a liquid rocket except for the use of the inlet, and can be considered an air-augmented rocket in its truest form. Because much more hydrogen is needed to cool the incoming air than is required for combusting with that air, the cycle is very fuel rich; however, the specific impulse produced is on the order of 1000 sec, and may be pushed as high as 6000 sec given the right technological gains. (Ref 2, 164).

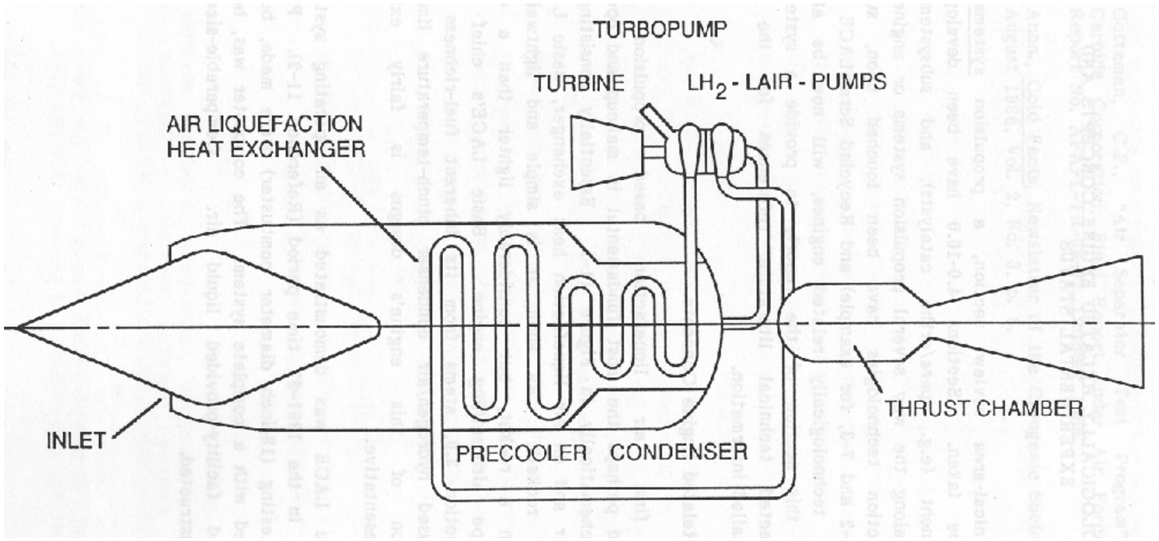


Figure 1: A Basic LACE Engine (Ref 3).

1.1.2 ACES

The Air Collection and Enrichment System (ACES) cycle shown in Figure 2 still utilizes air liquefaction, but instead of immediately using all of the produced liquid air, some or all is saved for use later on in flight. During early atmospheric flight, the oxidizer tanks can be refilled with liquid oxygen for upper atmospheric flight, where the atmosphere is thinner and there is less oxygen to be collected. This cycle does have some drawbacks; storing liquid air itself is a wasteful and inefficient process. Because liquid air is mostly nitrogen, tanking it means carrying a large amount of inert weight that will not aid combustion, raising overall system weight to the point where the advantages of the cycle disappear. However, through the use of a cryogenic rotary air separator, the lighter liquid nitrogen can be removed from the air and discarded or used to regeneratively cool the currently incoming air. This nitrogen can also be expelled through the engine’s nozzle as inert mass; while it does not combust, it can still aid in the production of thrust. The resulting tanked Liquid Enriched Air (LEA) is about 10% nitrogen. (Ref 2, 169).

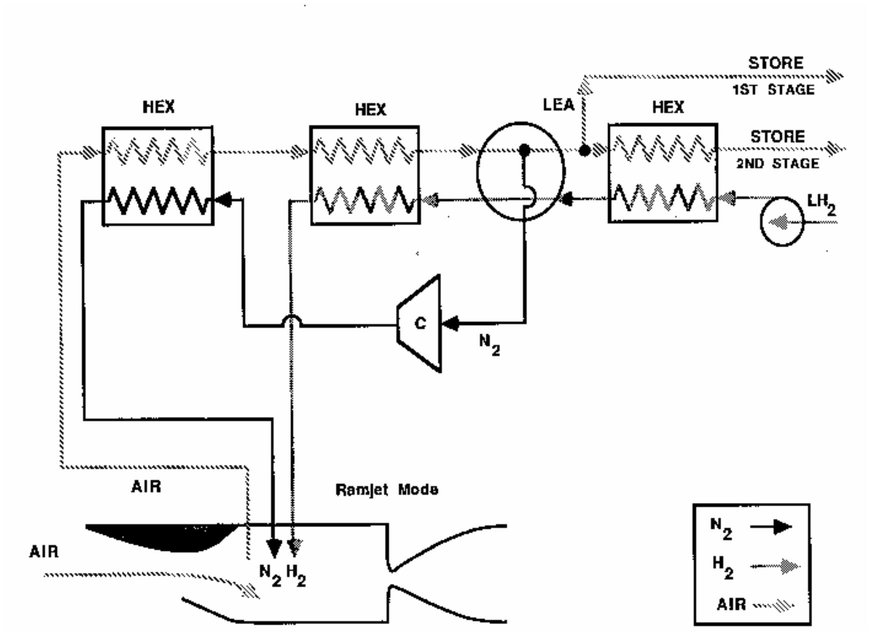


Figure 2: An ACES Schematic (Ref 3).

1.1.3 SuperLACE

As mentioned previously, the basic LACE is very fuel-rich. There have been numerous ways proposed to alleviate this problem and lean out the cycle. The SuperLACE concept combines three of these approaches to achieve this leaning out. It consists of a precooler and two condensers. The precooler is different in that it uses regenerative liquid air cooling in addition to hydrogen cooling. The liquid air enters this precooler after being compressed to high pressures in a pump, after the gains from converting the air to a liquid have already been made. Secondly, it uses a para/ortho hydrogen converter to make the hydrogen more effective as a heat exchanging fluid. This process will be described in detail in Section 1.2.1. A turbine expander is inserted between the first condenser and the precooler, and extracts energy from the hydrogen before returning it to the LH2 tank. This recycling of the hydrogen reduces the extra amount that must be carried to cool the air. Finally, the hydrogen itself is stored as its slush form. This partly solidified hydrogen provides extra cooling capability, and aids in the recycling of liquid hydrogen. The combination of these technologies is what enables the LACE cycle to produce I_{sp}s of the order of 6000 sec (Ref 2, 170).

1.1.4 RamLACE and ScramLACE

Air liquefaction was not the only promising concept to be studied in the 1960s. At this time, the ramjet and scramjet (supersonic combustion ramjet) combined cycles were also explored as enabling technologies for a spaceplane concept. Because these engines cannot provide thrust at low speeds or very high speeds, it was necessary to combine them with rocket engines. Many RBCC configurations were proposed, and among them were the RamLACE (Figure 3) and ScramLACE (Figure 4) concepts. During the mid-phase exploration of the various RBCC concepts, these two cycles were studied extensively, although the only design to be considered for detailed study was the ScramLACE concept.

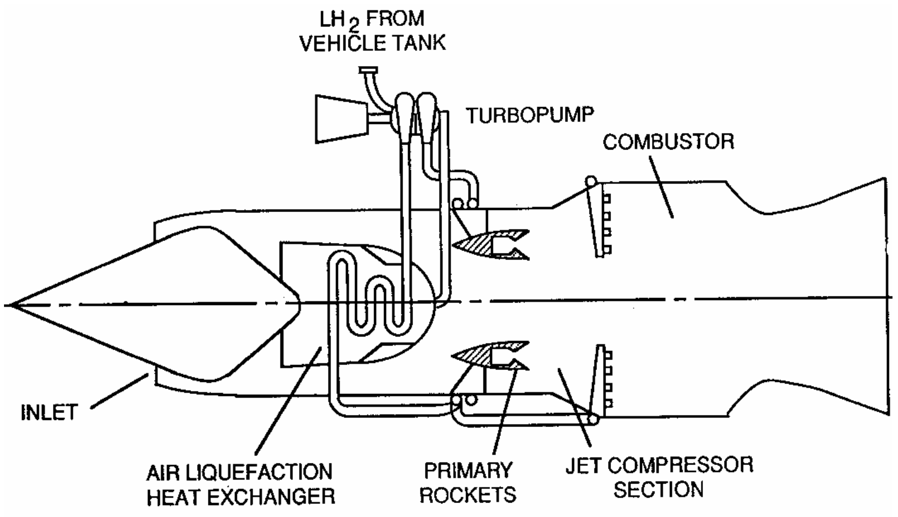


Figure 3: RamLACE Engine Schematic (Ref 3).

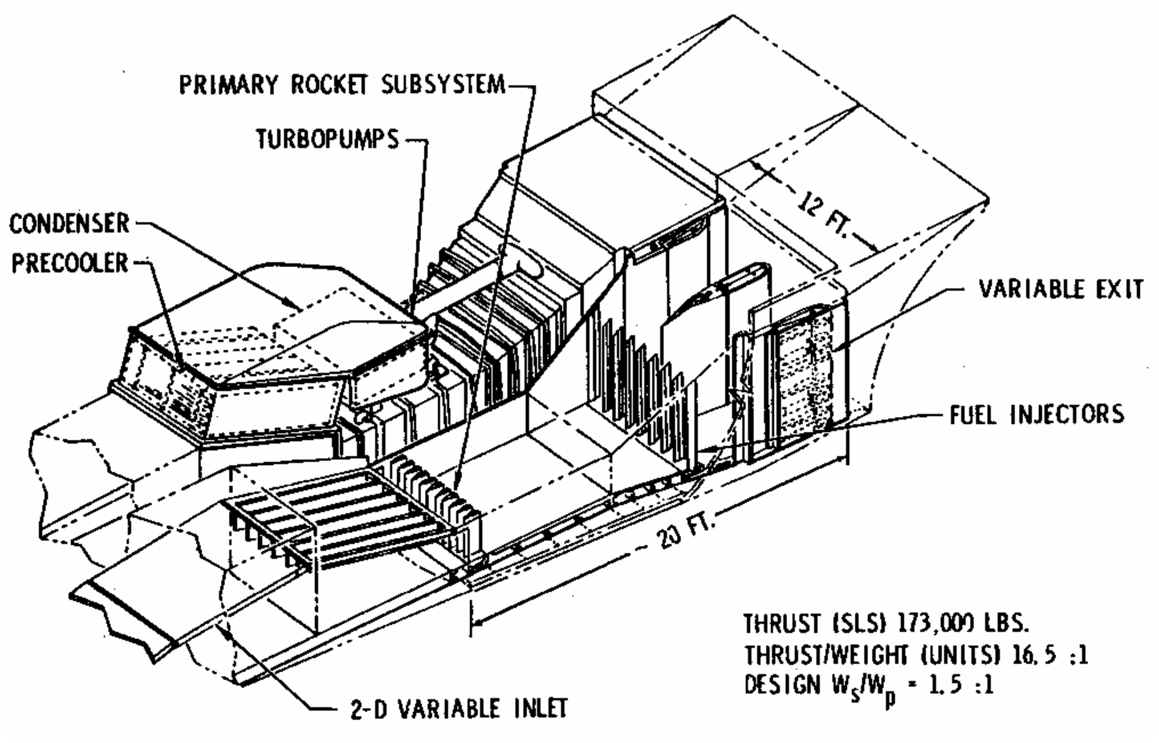


Figure 4: ScramLACE Engine Concept (Ref 3).

1.2 Enabling Technologies for Air Liquefaction

The development of the engine cycles outlined above also included the identification of technologies that would have to be developed in order to realize the performance gains assumed. Such technologies include catalyzing a para/ortho conversion in the liquid hydrogen as well as approaches for removing water from the air stream. These technologies include systems that will be modeled within the ALS, so a description of each technology and its benefits and costs is useful.

1.2.1 Para/Ortho Hydrogen Conversion

The hydrogen atom consists of a single proton and electron. While the fact that electrons have a spin parameter is well known, the protons of the hydrogen atom also spin. Therefore the hydrogen molecule, which consists of two hydrogen atoms, has two different configurations; the protons may be spinning in opposite directions (anti-parallel), or the same (parallel) direction. The former kind of hydrogen is known as parahydrogen, while the latter is orthohydrogen. Because there are basically three ways in which a binary molecule like H_2 could be structured as orthohydrogen, and only one way for it to be in its para form, H_2 at equilibrium is approximately 75% ortho and 25% para.

When liquid hydrogen is produced, this equilibrium is affected. In fact, at cryogenic temperatures the para form is favored, and the hydrogen will shift to this form. This is a very slow process normally. However, the process is exothermic, releasing heat and raising the temperature of the hydrogen liquid. Hydrogen producers have to provide extra refrigeration in order to combat this process, or the liquid hydrogen would boil off in a number of hours. Because all liquid hydrogen will essentially be supplied as 100% parahydrogen, it is possible to take advantage of the reverse reaction (para to ortho). This process is endothermic, and hydrogen converting to its ortho form will readily absorb heat from its surroundings. The applicability to air liquefaction is obvious; allowing hydrogen to shift from para to ortho form greatly enhances the ability of hydrogen to cool incoming air (Ref 6).

It seems as if this process should occur naturally without any need for technological development, and it does; however, the actual conversion of hydrogen is a very slow process normally, on a time scale that is unsuitable for air liquefaction purposes. Therefore a catalyst must be used to drive the para/ortho conversion at a suitable rate. These catalysts have been studied extensively in the air liquefaction industry. However, the facilities used by this industry are all ground based, and thus unconcerned with the weight issues of flight. Para/ortho catalyzers have been proven up to efficiencies of 75%. Such a shift in hydrogen properties can provide a “refrigerative enhancement effect” of about 1.3 (Ref 3, 6-4). This translates to a leaning out of the cycle, lowering the equivalence ratio by a factor of 0.8. System weight reductions are obviously made by this catalysis, but the catalyst itself adds to the system weight. The most promising of the possible catalysts is ruthenium (SiO_2,Al_2O_3), but previous efforts at creating a catalyzer capable of converting 1 lb/sec of parahydrogen to 75% ortho per pound of catalyzer failed (Ref 3, 6-6).

1.2.2 Compact Cryogenic Heat Exchangers

Although heat exchangers are very common and a number of different types exist, applying them for use in an aircraft system severely limits the choices available. Heat exchangers as flight hardware must be small, light, and reliable, all while delivering the high amount of performance needed for air liquefaction. As mentioned previously, the air liquefaction process can most simply be described in two phases: precooling and condensing. Systems can be more complex than this, of course.

Figure 5 displays a common temperature profile of the hot and cold fluids in a heat exchanging environment. The point on the graph circled, or more accurately the temperature difference between the two lines at that point, is known as the pinch temperature. This is the point in the heat exchanging system where the temperatures of the two fluids are most near to one another. This pinch temperature has important consequences on heat exchanger properties. Having a low pinch point (i.e. a small distance between the two lines) means that the system is very effective. This results in a reduction of the hydrogen necessary to cool the air to a liquid state, leaning out the cycle and helping alleviate a troublesome aspect of air liquefaction. However, it is also at this point that the heaviest heat exchangers are produced. A balance must be struck between the system weight and the hydrogen weight. Reasonably sized heat exchangers often have pinch temperatures around 10 to 30 R (Ref 3, 4-1).

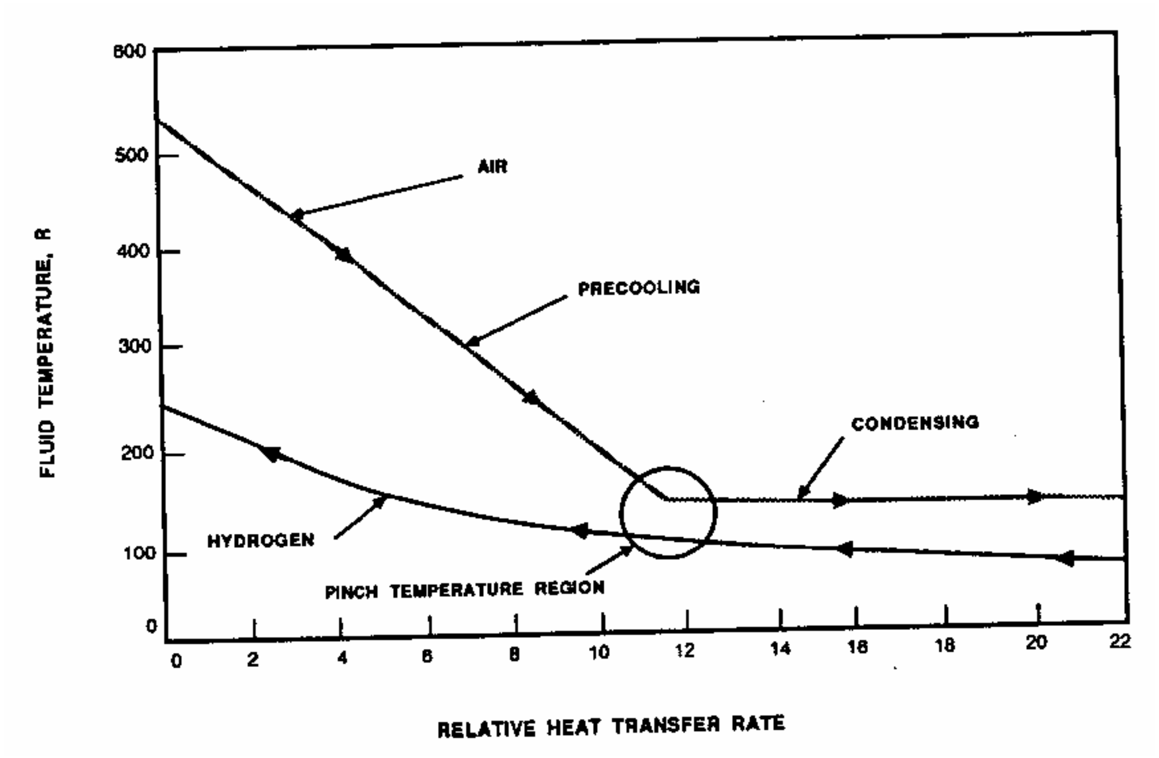


Figure 5: Standard Heat Exchange Setup (Ref 3, 4-2).

Heat exchangers for cryogenic applications are most often constructed of aluminum or stainless steel alloys. These metals have a reasonable coefficient of thermal conduction to enable heat transfer, can function within the range of temperatures necessary, and have a favorable density. Heat exchange may be carried out between the fluids in a variety of ways. The three primary schemes are parallel flow, cross flow, and counter flow. In parallel flow, the two fluids travel in the same direction, exchanging energy until they both reach some equilibrium temperature between their initial temperatures. Cross flow heat exchangers run the two fluid flows perpendicular to one another, offering a slightly more advantageous method of heat transfer. Finally, counter flow systems run the fluids in opposite directions, allowing heat exchange along the entire length; if the mass flows are in proper balance and the exchanger is long enough, the two fluids may swap temperatures almost completely by the time they have left the heat exchanger.

Because the two fluids must not mix during heat exchange, it is common to run the fluid with lower mass flow through tubes while the other fluid passes over the tubes.

These tubes may have fins which increase the heat exchange surface area but also increase the volume and subsequently the mass of the tubes. This increase in heat exchange effectiveness is offset by higher pressure losses and the increased ability for fouling agents to form. For precooling heat exchangers, both types of tubes are favored, while condensers mostly utilize bare tubes. Alternatively, a plate-fin system may be used instead of tubes. Plates are sandwiched on top of each other, and the fluids are run through channels in alternating layers. Plate-fin heat exchangers suffer the same drawbacks as tube-fin systems, however, although they are easier to manufacture.

1.2.3 Water Ice Fouling Alleviation

Because atmospheric air does not solely consist of oxygen and nitrogen, there is the opportunity for the fouling of heat exchanger surfaces as other substances liquefy and freeze before the air can even become a liquid. These fouling agents include carbon dioxide and argon, but by far the most prevalent danger is that posed by water in humid incoming air, especially during low atmospheric flight. Accumulation of water ice on heat exchanging surfaces causes two problems; firstly, it alters the conductive properties of the surface, and secondly it can block the mass flow of the air and degrade performance. The amount of liquid air produced will drop, and engine performance may suffer so badly that the engine can fail completely. Even when the water is removed from the air stream, there may still be parasitic weight issues unless the water is ejected from the aircraft completely (Ref 3, 5-7).

There have been many approaches to solving the fouling problem, many of them unsuccessful. The successful approaches include cyclic de-icing, snow formation, glycol injection, and liquid condensation. Cyclic de-icing involves turning off the air liquefaction system for short periods of time while the tubes are heated. While it performs as needed, the additional heat exchanging capacity, valving, and ducting adds significant weight to the system. In snow formation, the water in the bulk air stream is induced to freeze before entering the primary heat exchangers. This method still requires extra hardware to actually remove the "snow", and more space between the heat exchanger tubes in order to prevent the system from seizing up. One of the more promising approaches is to inject ethylene glycol as either a spray or in droplet form to prevent the water from freezing as readily on heat exchanger surfaces. This method has been experimentally tested and has provided good results. Finally, the air may be cooled to high humidity levels so that the water can be removed as a liquid from the air stream. Experimental work with this method is not as developed, but initial analysis is encouraging.

Liquid condensation and glycol injection may be combined to produce an even more effective method for fouling alleviation. Researchers from Marquardt studied this problem back in the 1960s when the air liquefaction problem was first explored. They tested an experimental system and successfully removed enough water to prevent the system from shutting down (Ref 3, 5-13).

2 Design Problem and Methodology

Air liquefaction has once again become of interest to NASA and the U.S. Air Force. Building on the knowledge obtained back in the 1960s when the problem was originally formulated, a new approach for modeling the system is desired. This approach will take advantage of the development of system integration software developed in the 1990s. Using the practices of Multidisciplinary Design Optimization (MDO), these software tools can combine a number of different contributing analyses that have historically been “owned” by one group or another within an organization. By allowing various codes to be “wrapped” and connected to each other, one can rapidly explore a design space and find the best design to meet a particular goal or set of requirements.

The goal here is to model an Air Liquefaction System (ALS) consisting of four heat exchangers and two water separators. The technical points of contact for the development of this system are Larry Hunt of SAIC, and Jeff Robinson and John Martin of NASA LaRC. An outline of the system can be seen in Figure 6. Air enters the ALS after traveling through the vehicle inlet. Conditions are prescribed to the ALS model after this inlet, and the air travels through the system being progressively cooled by the heat exchangers. These heat exchangers are of the counter-flow design, and are arranged in a tube bank architecture such as that seen in Figure 7. In the initial stages, the air is only cooled to the point where water can be removed from the air stream utilizing the water separators. After the first water separator, an atomized ethylene glycol spray is injected into the air flow. This glycol travels through the second precooler and is removed along with the water in the second water separator. Its purpose is to prevent the water from freezing and modifying the saturation properties of the humid air, thus making the removal of the water much easier.

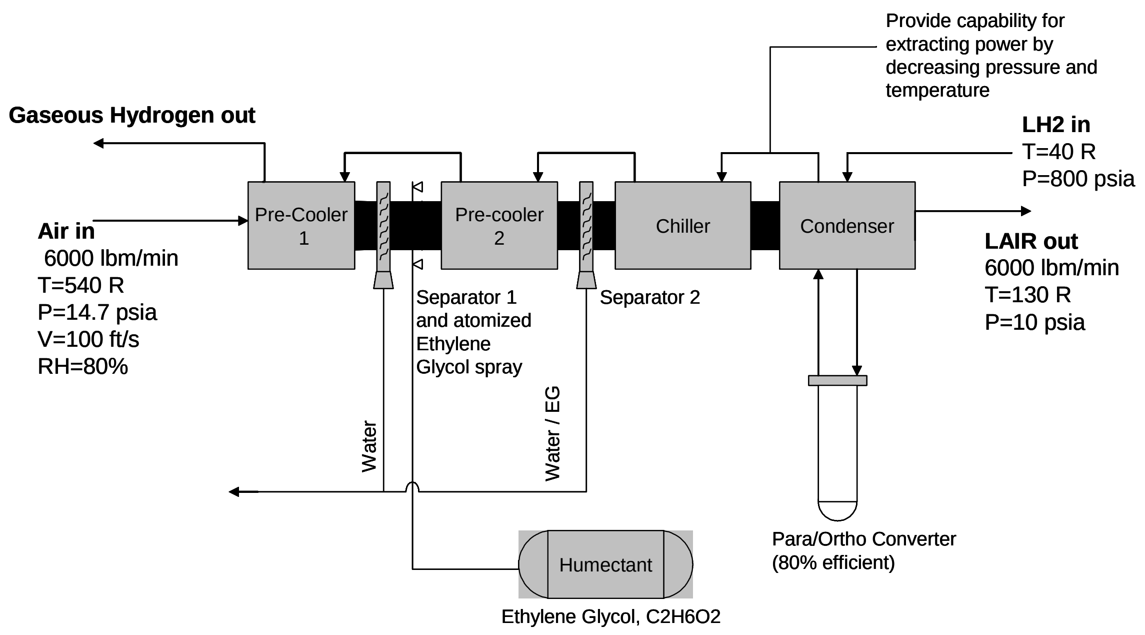
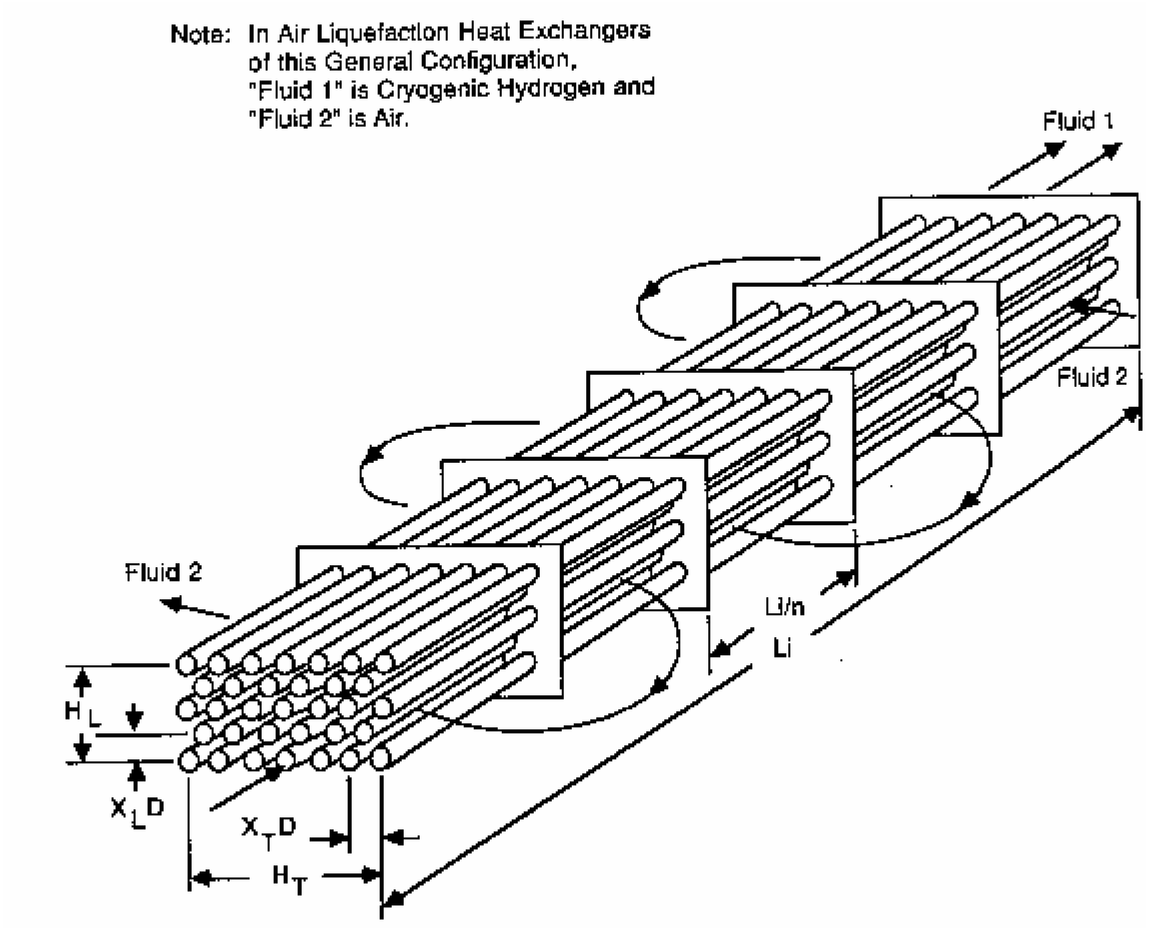


Figure 6: The Air Liquefaction System.



Note: In Air Liquefaction Heat Exchangers of this General Configuration, "Fluid 1" is Cryogenic Hydrogen and "Fluid 2" is Air.
Fluid 1
Fluid 2
Fluid 2
Li/n
Li
H L
X L D
X T D
H T

Figure 7: A Typical Tube Bank Heat Exchanger (Ref 3).

Properties specified after the inlet include the mass flow, temperature, pressure, velocity, and relative humidity of the incoming air. These properties are fed forward through the system, with the output from each successive component becoming the input for the next section. The outputs will depend on the inputs from the "cold" side, or the liquid hydrogen properties as it enters the heat exchangers. The liquid hydrogen only travels through the heat exchangers, and its mass flow may be modified at each point by removing some mass and either returning it to the tanks or feeding it to the combustion chamber. The properties of the liquid hydrogen are specified at the inlet to the condenser stage. These properties include the temperature and pressure of the liquid hydrogen.

The goal of the system is to obtain liquid air by the end of the cycle, at a temperature of about 130 R (-330 °F) and a pressure of 10 psia. The mass flow rate of liquid air should be 4 to 5 times that of the liquid hydrogen required to cool the air to its liquid state. These goals must be met while keeping system weight (and thus costs) down.

MDO practices will be used to reach this goal. The components will each be designed separately so that they can be used alone or in tandem to model the ALS itself. The components will then be assembled within a modeling environment so as to simulate the entire ALS at once. This will be accomplished on two separate software environments: Phoenix Integration's ModelCenter and Modelogics, Inc. Model Engineer. Both pieces of software are designed to allow a user to "wrap" user codes, whether legacy or newly devised, and combine them to make a larger system. ModelCenter is widely used within Georgia Tech's Space Systems Design Lab and also within NASA and industry. Recently Model Engineer has become involved in a number of Air Force and other government contracts. It boasts a large library of predesigned components and is especially well-suited to tackling a thermal systems problem. However, its optimization capabilities are not as developed as ModelCenter's. Additionally, each platform uses a different method of "wrapping", thus requiring that the components be coded in two different languages.

2.1 Model Engineer

Model Engineer relies on Microsoft's COM framework for computers. This makes it possible to use Model Engineer's components on any computer using the Windows operating system, just as one can open an Excel file on any computer with Office installed. This is achieved using the highly flexible Visual Basic language. New codes may be created directly using Visual Basic and the utility Object Engineer, or legacy codes may be wrapped using DLLs. FORTRAN codes, Excel spreadsheets, any many other types of codes can be directly wrapped in Model Engineer and incorporated into a model. The toolkit comes supplied with several libraries created by Modelogics, Inc. The most useful libraries are the Thermal Systems, Data Viewing, and Data Flow libraries. These features will be described in detail (Ref 4).

2.1.1 Object Engineer

Object Engineer is a utility included with the Model Engineer toolkit. It simplifies the component creation process with a series of input pages. The Control page is where the name and type of the component is defined, and is where embedded components are added. If there is a legacy code that needs to be wrapped, it is possible to create a DLL in a program like Visual Fortran or Visual C++. This DLL is then referenced in the Control window and is fully available to the component. There are also a number of references already provided in the Model Engineer toolkit that may be embedded in the component.

The Property window is the heart of the component creation process. It is here that each variable of interest is defined, typed, and bounded. There are two basic types of variables, not considering types such as integer, double, etc. Data flow variables consist of an input and output variable, and thus exist to pass information in and out of the component. When building models, links can be created between components using these flow variables. For example, the mass flow through a heat exchanger would be modeled by a data flow variable. In addition, there are static variables; these consist of simple functions that compute a single property. Variables can be grouped, and placed on different pages in the eventual windowed model created by Model Engineer. Object

Engineer will automatically use the variable definitions to create property pages in Visual Basic, eliminating most of the grunt work of coding the components.

The final window is the Function window. Any variable that requires calculations will be defined as a function within the Visual Basic pages created by Object Engineer. Using the Function window, the variables necessary to make those calculations will be automatically provided to the function. It is simply a matter of selecting the variables defined in the Property window and checking boxes next to the variables required to make calculations.

It is not necessary to define the component in Object Engineer, although the utility is required to actually build it. The three windows previously discussed are used to create an MS Excel file that lists all of the variables and their properties. A user sufficiently familiar with the component he is creating and Model Engineer itself can define a component entirely in Excel, and often much faster than within Object Engineer itself. This Excel file, a comma separated value file, can be imported into OE and used to create the Visual Basic files that will be used to complete the component creation.

The files created include class and control forms, as well as the forms that a component user will see when he uses a component. These files are manipulated using Visual Basic. Most of the work needed to create a component has been done by Object Engineer; essentially all that is required is to define the functions that will calculate the variables of interest. One drawback to this approach is that every time a variable is added to the program, it is necessary to rebuild the component in Object Engineer and then rewrite all of the functions over again. Often this is just a matter of cutting and pasting, but this can be a time consuming and frustrating process. Presumably, a strong knowledge of Visual Basic would prevent the need for rebuilding components every time it is updated, but for the casual user it is much easier to rebuild. It is a testament to the flexibility of Model Engineer that a novice user of Visual Basic can easily create components.

2.1.2 Thermal Systems Library

Although the user is in no way limited to the components and libraries already created by Modelogics, a number of these components are available. The thermal systems library was created as part of a Modelogics contract with the Air Force, but is available to all Model Engineer users. It includes common thermal components such as cross flow, counter flow, and cooled panel heat exchangers. It also consists of turbines, compressors, condensers, feed lines, among others. These components are clearly useful in designing the Air Liquefaction System, but as will be seen later there are advantages to creating custom components to fulfill the ALS requirements. These components are added to a Visual Basic form that is created by starting a new Standard EXE file in Visual Basic. By simply including a library and dragging components into the active window, a model can be created.

Upon first glance, the cross flow heat exchanger is perfect for purposes of designing the ALS model. Its data flow variables include mass flow, temperature, pressure, and enthalpy. It is designed to model a number of different working fluids, using Model Engineer's Map Tool. This tool is used to conduct table lookups of various fluid properties. The input values for the hot and cold sides of the heat exchanger are specified, as well as a desired heat exchanger efficiency.

For the final system model, however, it was decided that new heat exchanger components would be created. This was done for several reasons, one being that it was desired to gain an understanding of heat exchangers rather than utilizing a pre-existing "black box". Additionally, the provided heat exchanger component consists of functions and inputs that are not used in the ALS model as specified in the problem statement. A much more streamlined heat exchanger could be built that would execute faster and could be modified as needed. Finally, the fluid databases native to Model Engineer are not best equipped to handle liquid air or cryogenic hydrogen. This user-created component would also expand the capability of Model Engineer in some ways, as it is provided to Modelogics as part of the agreement that allowed Georgia Tech to obtain the software.

2.1.3 Data Viewing Library

The Data Viewing library helps make sense of the possibly large amount of data that is produced by a model. Using the Schematic Viewer (or S-Viewer), a pictorial representation of the model can be displayed while it executes. This picture is created in Microsoft PowerPoint and saved as a Windows metafile (*.wmf). By editing the S-Viewer's properties, any variable from any component can be linked to the picture, and the current value of that variable will be displayed on the picture at every point in time. The units may also be displayed. This functionality can be very useful during model execution, allowing a user to visually understand the direction a model is taking and identify points in the model of interest.

Model Engineer also provides the capability to interact with Microsoft Excel and PowerPoint using the Data Viewing library. The ReadXcells, WriteXcells, and PasteViewtoPPT components all open the appropriate MS Office application and can paste or copy whatever data is needed. In this way, the input variables may be defined in an Excel worksheet and read by the component. Output variables can be similarly written to an Excel sheet where they can be further manipulated. The pictorial representation of the model in the S-Viewer can be pasted into a Power Point slide making it easy to create visuals for a presentation or paper. Overall, it is a much more visual application than using a standalone code or some other modeling approach.

2.1.4 Data Flow Library

The Data Flow library makes it possible to build models out of individual components. Its primary components include the Connector Arrows which pass data between components' data flow variables. When the arrow is dragged from one component to another, a connection definition window will open; if the components are designed for modularity well, Model Engineer will automatically connect the outputs of one component to the appropriate inputs of the other component. Dragging an arrow from one heat exchanger to another, for example, will neatly match up the mass flow, temperature, pressure, and enthalpy data flow variables. By defining groups of variables, it is possible to automatically match variables like hot side (air) properties to other hot side properties, and similarly cold side (hydrogen) properties. This can all be controlled from the arrows' properties window summoned by right clicking on the arrow itself on the Visual Basic form.

The Data Generator is one of the most important components of this library. It drives the model forward and defines the execution order for each component. It can also specify points in the model where iteration should occur. When the model executable is created, this component is the button that begins the execution. It can be configured to run in series or step mode; in step mode, the model will execute only once. In series mode, a specific input variable can be changed by some increment for a specified number of iterations.

Although Model Engineer's optimization capability is somewhat limited, it does have a limited functionality. The Data Flow library has components that can iterate on specific variables to optimize another variable. These components include the RFcontroller and Interval Halving. The RFcontroller uses the Reguli-Falsi method of finding an optimum for a single variable and response (Ref 4). The Interval Halving method examines the response for the range of a single input variable and halves that range until it finds the optimum point. It will only function on a monotonically increasing or decreasing response. For advanced optimization, it is better to use a program like ModelCenter.

2.2 ModelCenter

ModelCenter is a product of Phoenix Integration and is a package designed to allow the integration of various contributing analyses and then conduct analysis on them. User codes can be accessed in ModelCenter through the use of various "wrappers", depending on the type of input and output the program uses. This includes any file based program no matter the language used, and MS Excel files. More complex wrapping can be accomplished through the use of scripting; many scripting languages are supported including Java and Visual Basic or VBA. New codes can easily be written within

ModelCenter itself, although it is generally as easy to write a file-based input/output code in the programming language of preference.

2.2.1 FileWrappers

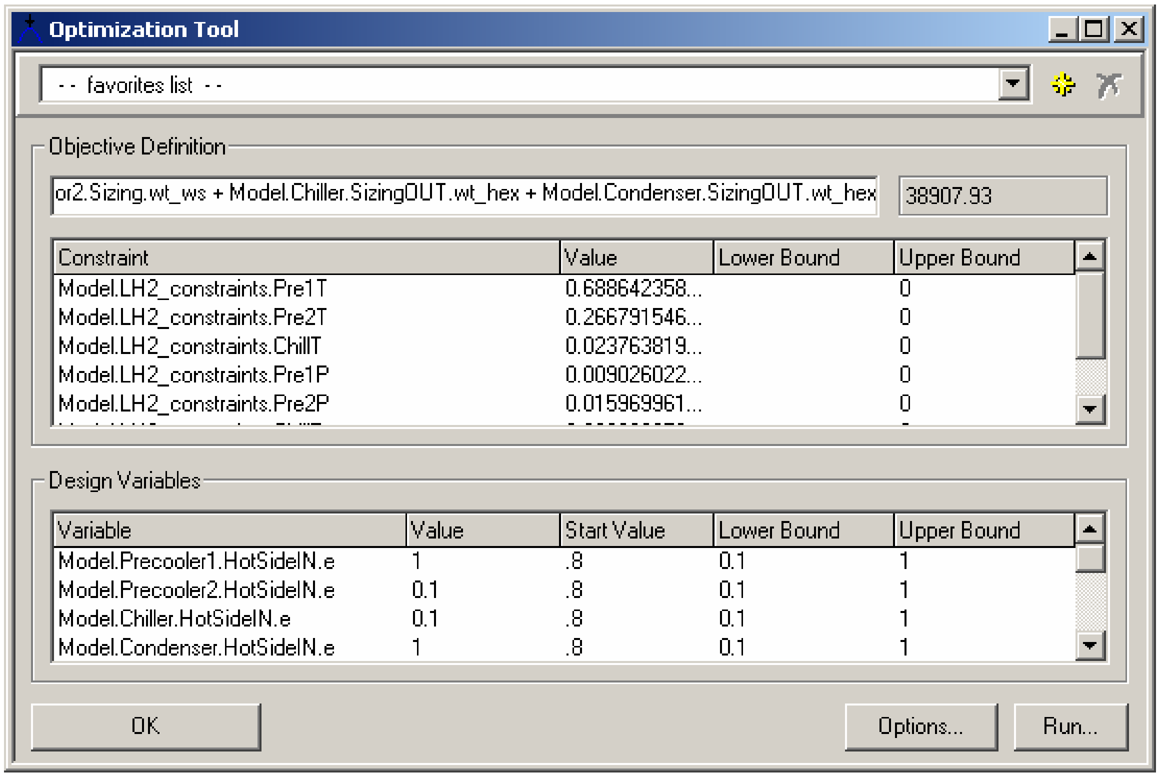
By far the most common kind of wrapper is the fileWrapper, which instructs ModelCenter where to look in a file for input and output information. A basic fileWrapper consists of three sections; input, execution, and output. The input section contains all the information needed for ModelCenter to write the code's input file. Using a template file, ModelCenter can be instructed where in the file certain variables are to be stored. Using ModelCenter's interface, a user can enter data of any supported type (such as integer, double, or string) within bounds set by the wrapper's author. When the wrapper is executed, the template file is opened and the wrapper uses indices to tell where the data should be placed. Different template files can even be specified if a program's input file changes considerably for different analyses.

The wrapper then executes the code of interest. Generally the executable, as well as any input, template, output, and other necessary files must be stored in an Analysis Server directory. Analysis Server is the portion of ModelCenter that manages fileWrappers and serves wrappers and executables to any user who can connect to the server. In this way distributed computing becomes very simple; anyone who can connect to an Analysis Server may run the analyses therein.

Once the file has been executed, the wrapper can open any resulting output files and read the data inside. By means of the same indexing as used to specify input variable locations, ModelCenter will read the output and display it within the user interface. Multiple output files can be handled by adding multiple output sections in the wrapper. The data obtained from the output can be graphed, studied parametrically, or even optimized upon using the tools available in ModelCenter.

2.2.2 Optimization

Optimization is a relatively simple matter in ModelCenter. By creating a model, either out of single components or multiple ones working as a system, any variable can be used as an objective function, constraint, or design variable within an optimization problem. If the objective of interest can be calculated from data available from the components, it is a simple thing to create a script to calculate an overall evaluation criterion. The optimization window has three areas of interest. The objective function is defined in the first box in Figure 8. Values from the model can simply be dragged into the box to define the optimization criteria. Additional variables can be dragged in to add them to the current function. For example, if one wanted to minimize the system weight of multiple heat exchangers, the individual weights from each heat exchanger component could be dragged into the window and ModelCenter would automatically add them together.



or2.Sizing.wt_ws + Model.Chiller.SizingOUT.wt_hex + Model.Condenser.SizingOUT.wt_hex   38907.93

Constraint	Value	Lower Bound	Upper Bound
Model.LH2_constraints.Pre1T	0.688642358...		0
Model.LH2_constraints.Pre2T	0.266791546...		0
Model.LH2_constraints.ChillT	0.023763819...		0
Model.LH2_constraints.Pre1P	0.009026022...		0
Model.LH2_constraints.Pre2P	0.015969961...		0

Variable	Value	Start Value	Lower Bound	Upper Bound
Model.Precooler1.HotSideIN.e	1	.8	0.1	1
Model.Precooler2.HotSideIN.e	0.1	.8	0.1	1
Model.Chiller.HotSideIN.e	0.1	.8	0.1	1
Model.Condenser.HotSideIN.e	1	.8	0.1	1

Figure 8: ModelCenter Optimization Window.

The next box in the window describes the constraints on the problem. Constraints may be defined in the window by dragging any variable into it, and setting upper and lower bounds that variable must meet. In the options window, constraint criteria may be altered. The conditions upon which a constraint is violated or met can be changed. If different constraints have highly different values, it may be beneficial to have ModelCenter normalize them. Alternatively, such alterations may be made within a script.

Variables can be squared and normalized within a script, making them easy to define. For instance, if the output temperature of a component was desired to remain at a certain point, that desired temperature could be fed as an input variable to a script. The actual temperature would also be fed as an input variable. By subtracting and squaring the difference between the two, adding that output to the optimizer, and putting an upper bound of 0 on the constraint, the optimizer will strive to make sure the actual and desired temperatures match. This allows an equality constraint to be modeled as an inequality constraint which can be much more easily handled by many optimizers.

The final portion of the main optimizer window is the design variable window. Again, by simply dragging model variables into the window, it is possible to define all the variables that can be changed during optimization. A starting value for each variable can be chosen, or if left blank the optimization simply begins at the current point in the model. This can be useful if a design space is finicky and needs a good starting point, or if an optimization stops before being fully completed. The values that the optimizer is allowed to set each variable at can also be specified. Therefore the optimizer can be stopped from causing a problem if certain values make no sense or would cause a program crash.

The results of the optimization are stored in a table that can be exported as a comma separated value file, or viewed from within ModelCenter. In fact, ModelCenter will plot any combination of variables automatically by choosing the independent variable, and adding as many dependent variables as desired. This greatly simplifies the understanding and visualizing of the process of optimization. Plots can easily be created for any purpose.

2.2.3 Link Editor

The link editor is what enables a user to build a model from disparate components. When an output from one component must be passed to another, this is done by dragging the output variable on top of the input variable in the link editor screen. A connection is created, represented by a black line between the two variables. Breaking the link is possible by selecting a variable and clicking the break link button. Links that feed forward in a system are preferred from a Multidisciplinary Design Optimization (MDO) perspective, but it is possible to create feedback links. When the model is executed, it will iterate until all links are updated and agree, although this feature can be turned off.

2.3 The ALS Model

The ALS model created here utilizes several of the concepts explained in the introductory section to air liquefaction. As stated, it consists of four heat exchanging sections and two water separators. These components were designed so that they could be executed within Model Engineer and ModelCenter, as well as exist as standalone components. The ModelCenter components are written in C++ and utilize basic text file input and output. These components can therefore be compiled as executables and run outside of any modeling environment, or wrapped and executed from ModelCenter's Analysis Server. Both C++ and Visual Basic serve well as programming languages; C++ serves especially well due to its heavily object-oriented approach. The Model Engineer components use the same basic functions and calculations, but are coded in Visual Basic using the Object Engineer utility. The use of these components is explored explicitly in the User's Guide of Appendix A, but the general function of each is outlined below.

2.3.1 Heat Exchanger Components

Whether executed in C++ or Visual Basic, the heat exchanger components conduct the same analyses based on the same equations. The heat exchangers used in the ALS are of the counter flow design, with a tube bank architecture. Air enters the heat exchanger and is exposed to pipes filled with flowing liquid hydrogen. The input properties of both the fluid streams (referred to as the hot and cold sides) are specified by the user, and the component returns the output properties of both streams, in addition to heat exchanger properties such as size, weight, and relative cost. The inputs are defined in the file hx.inp, written to the output file hx.out, and the program is executed using HX.exe.

Heat exchange within the component is governed by the basic equations of heat transfer. Due to the law of energy conservation, any heat lost by one fluid will be gained by the other (discounting losses to the surroundings). This is expressed in Equation 1:

$$\left[\dot{m} C_p \Delta T\right]_{HOT} = \left[\dot{m} C_p \Delta T\right]_{COLD} \tag{1}$$

This equation assumes that neither fluid is changing state; if this is the case, then the heat lost or gained by one fluid is equal to

$$\dot{Q} = \dot{m} \Delta H_v \tag{2}$$

where H_v is equal to the heat of vaporization of the fluid. It is assumed that no other state changes will be made other than the liquid/gas transition. The specific heat C_p is normally a function of temperature for a gas; for a liquid, it can be assumed constant. When the heat exchanger evaluates C_p of a gas for calculation purposes, it does so using JANNAF curve fits. This C_p is evaluated at some average temperature.

The output temperature of the air is defined by inputting a desired effectiveness into the component. This takes into account the difference in initial temperatures of the two fluids, and modifies the air temperature by some percentage of that difference. This is expressed in Equation 3.

$$T_{2_{HOT}} = e_{HOT}(T_{1_{HOT}} - T_{1_{COLD}}) \qquad (3)$$

Based on this required temperature change, the heat needed to be extracted from the air stream can be calculated using the relations above. This heat is then added to the hydrogen stream and its output temperature can be calculated. This temperature can also be related as a cold side effectiveness based on the initial temperatures of each flow.

An effectiveness of 1 for either side would technically require an infinitely long heat exchanger. In reality, it is possible to get very close with a very long heat exchanger, so there is no error returned by the program if the effectiveness is input as 1. However, it may be impossible in any case to achieve any desired effectiveness due to insufficient mass flow of liquid hydrogen. If this is the case, the program will find the cold side effectiveness to be over 1. This is impossible, as it indicates that the temperature of the hydrogen has exceeded that of the air. Once the hydrogen reaches the same temperature as the air, it will no longer result in any heat transfer. Therefore, the program calculates the amount of possible heat that can be absorbed by the hydrogen stream. This heat is extracted from the air, and then the actual hot side temperature and effectiveness are calculated and written to the output file. Additionally, this condition triggers an error flag that is reported in the output file.

These hot and cold side properties help define the size and weight of the heat exchanger. Based on the velocity of the incoming air stream, its density, and the mass flow, a certain cross-sectional area will be required. The velocity and mass flow are inputs; the density is calculated using perfect gas relations or constant liquid densities depending on state. To help size the heat exchanger in the length direction, the diameter of the hydrogen tubes is specified, as well as a material. Currently, aluminum and stainless steel are the only options, and serve well as heat exchanging material. Based on the conductive properties of the material, a certain amount of surface area will be necessary to accommodate the heat transfer between air and hydrogen. This surface area is divided by the number of tubes present to find the surface area of a single tube. Since the diameter is specified, the length of a tube can be calculated from this required surface area. Figure 9 shows a common spacing; there is 2.0-2.5 times the diameter of a tube between each tube. Although normal heat exchangers may have much more tightly packed matrices, this spacing helps design against any water freezing that may not be removed by the separators.

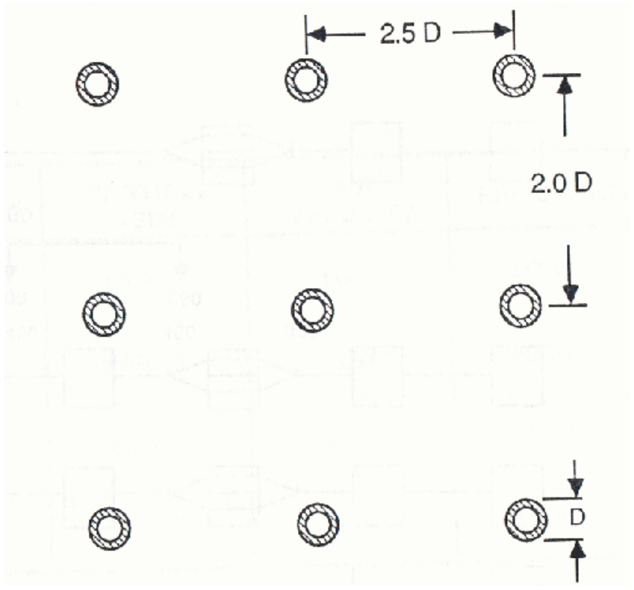


Figure 9: Heat Exchanger Tube Spacing (Ref 3).

Given the material and volume of the heat exchanger, a core weight can be calculated. This core weight represents the mass of a volume that is filled solid with the material. Given this core weight, the total weight of the heat exchanger can be found (Ref 9, 244). This guide is an industry standard even 20 years after its initial publication. Using the same source, relationships for cost, reliability, and development risk are found. These relationships unfortunately do not include units, but optimizing upon them can be beneficial. As they are generally functions of weight, however, it is probably just as useful to optimize upon weight.

An ALS heat exchanger can also utilize a para/ortho conversion catalyst to improve heat transfer efficiency at the cost of the extra catalyst weight. This functionality is accessed by using a switch variable for the para/ortho catalyst. When the switch is on (has a value of 1) the initial heat transfer is conducted through the conversion process rather than raising the temperature. The incoming hydrogen must be liquid for the para/ortho shift catalyst to be used.

2.3.2 Water Separator Components

The two water separators in the ALS model are essentially the same; however, the second separator must take into account some mass of ethylene glycol in the stream. The basic idea behind the water separator is that the air enters the separator at a very high relative humidity. This is achieved by cooling the air in the earlier heat exchanging sections. The water droplets moving in the stream, not yet cold enough to freeze, hit vanes within the separator. While the air hitting these vanes continues through the system uninterrupted, the water is stopped and is removed from the system due to gravity. There are several issues that must be dealt with in order to model a water separator.

The first is that the humidity of the system is defined at the inlet to the ALS itself, not at the separator entrance. As the air goes through the precoolers, its temperature and pressure drop, altering the relative humidity of the air. However, the absolute humidity in pounds of water per pounds of air remains constant. The separator therefore takes in the reference temperature, pressure, and relative humidity, as well as the current temperature and pressure. The reference absolute humidity is calculated from the other reference properties, and this absolute humidity is converted to a relative humidity for the air entering the separator. The equations governing this process are outlined in Equations 4 through 6 (Ref 1, Ref 8).

$$HR = \frac{p_v}{p - p_v} \tag{4}$$

$$p_v = 8.8624x10^{-2} \exp\left[\frac{17.67\left(\frac{T}{1.8} - 273.15\right)}{\frac{T}{1.8} - 30}\right] \tag{5}$$

$$p_{vTdew} = RH * p_v \tag{6}$$

HR stands for absolute humidity, while RH stands for relative humidity. The pressures given are measured in psia, while the temperatures are in degrees Rankine.

For the separator to remove any water, the relative humidity must be 100%. For any air stream considered "humid", this will generally be the case by the time the air reaches the separator. The separator will not remove all water from the air, of course. One of the system inputs is the efficiency of the water separator. This value is by default 85%, a standard value for a typical water separator and well within obtainable limits. This default should not be changed under normal circumstances.

The bigger problem with separating water occurs at the second water separator. At this point the air has been cooled twice, and may be at the point where the condensed water will freeze. There are separators mentioned in Section 1.2.3 that remove frozen water from a system; however, that solution is not desired here. The first separator is designed to remove liquid water from the air, and from a robustness and reliability standpoint it is better to have two separators that are essentially the same design. Therefore, the solution of adding ethylene glycol ($C_2H_6O_2$) to the air is modeled. This ethylene glycol is added to the flow as an atomized spray before the air enters the second precooler. There, it has very little effect on the heat exchanging properties of the mixture except to add to the mass flow.

However, it has multiple effects on the operation of the water separator. Ethylene glycol is entirely miscible in water. The model therefore assumes that the atomized spray will mix perfectly with the water droplets that manage to escape the first water separator. This will lower the freezing point of the water as outlined in Table 1. The heat exchanger is protected from having ice water fouling, and the water separator can remove the water from the air. The ethylene glycol is removed as well, as it has mixed perfectly with the water droplets. Of course, because of separator efficiency, some water and glycol will remain in the stream, but it should not contribute significantly to fouling.

Table 1: Effect of Ethylene Glycol on the Freezing Point of Water (Ref 7).

% Ethlyene Glycol	Freezing Point of Water (R)
0	491.67
10	484.67
20	479.67
30	464.67
40	449.67
50	429.67
60	404.67
70	399.67
80	409.67
90	439.67
100	469.67

The ethylene glycol has a smaller, secondary role in that a water/glycol mixture has different vapor pressure properties than plain water. According to Raoult's Law (Ref 5), the vapor pressure of a mixture is defined by Equation 7, where χ_i and p_{vi} represent the molar fraction and vapor pressure of the compounds comprising the mixture.

$$p_v = \sum \chi_i p_{vi} \tag{7}$$

Ethylene glycol has a vapor pressure of 0.00135 psia at room temperature; therefore, adding it to the water droplets causes the vapor pressure of the mixture to be lower than that of water by itself. This has the effect of raising the boiling point of the mixture. The air will reach its saturation point at a higher temperature and water will condense out of the stream much easier. This is generally not an issue since the water should have condensed already in the first separator, but it is an added benefit to consider.

The water separator also has weight, sizing, and cost relationships. These relationships come from the same source as the heat exchangers (Ref 9, 288). These relationships are based on smaller scale rotary water separators, so they may not be quite as accurate as desired.

2.3.3 Model Engineer ALS

The components were created in both Visual Basic and C++; it is the Visual Basic components that go into the Model Engineer ALS. The model is created by compiling the components into OCX files, then starting a new EXE Project in VB. The components are simply dragged onto the form, and then connected using the Model Engineer Data Flow library. There are two approaches to making these connections; both the hot and cold side inputs and outputs can be linked, forcing the model to execute until every side is matched properly. This requires some method of working around the connecting restrictions, as there can only be one directional link from one component to another. By adding the Splice component, the cold side data is passed through the component and then to the next heat exchanger. This set up can be seen in Figure 10.

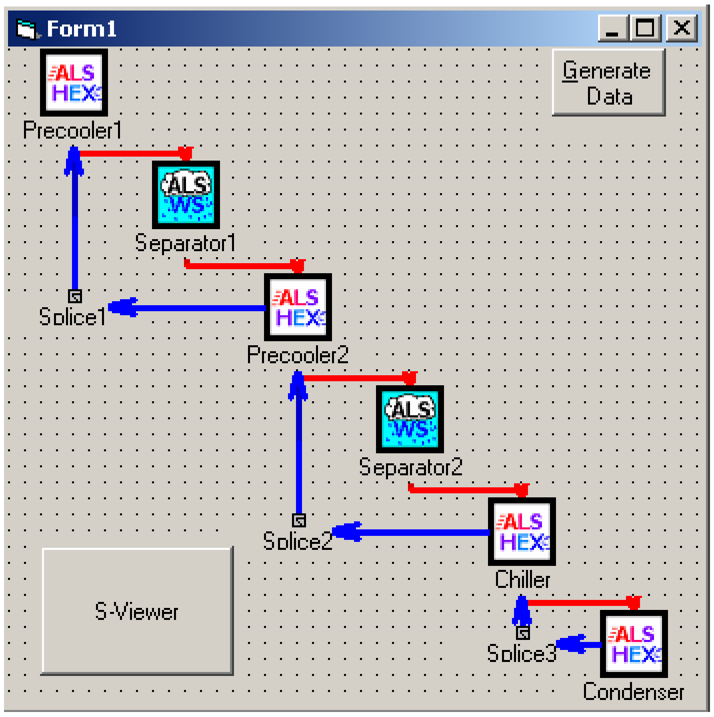


Figure 10: Feedback Version of Model Engineer ALS.

This “brute force” method of executing the ALS runs in a reasonable timeframe, coming to a possible solution in less than a minute given reasonable inputs. However, this solution is not guaranteed to be optimal, and may in fact be far from optimal. Furthermore it is possible, given poor inputs, for the system to become increasingly more divergent from a converged solution.

A better alternative from an MDO perspective is to not create these cold side feedback links. Model Engineer is not natively suited to MDO practices, but it is possible to set up a Fixed Point Iteration scheme by utilizing Excel files and the ReadXcells and WriteXcells components. Initial values for all of the hot and cold side inputs are set within the Excel input file. Model Engineer reads these inputs and executes the model once, returning the outputs of each component to the Excel file. The Excel sheet is able to take those outputs and return them as new inputs to a new execution. Using the Data Generator, the model is told to repeatedly run through this process some set number of times. The current state of the model and how well the inputs match the outputs is tracked through the Excel file. Unfortunately, it is not possible to tell the model to stop when convergence is obtained. An appropriate number of model executions must be determined from trial and error. Luckily, even when unnecessary executions are made, they do not take very long; the components execute very quickly. The model for this setup can be seen in Figure 11.

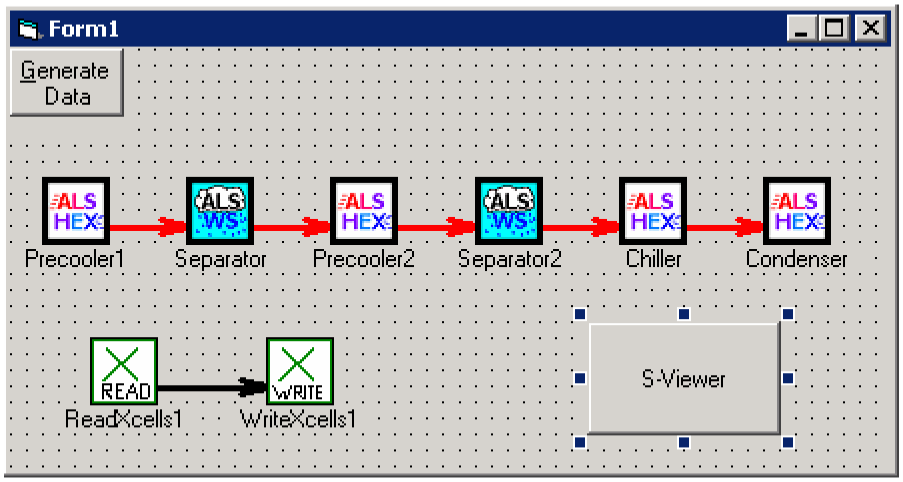


Figure 11: FPI Version of Model Engineer ALS.

Comparing the two approaches, it is found that the feedback method tends to work slightly faster due to the inevitable system slowdown caused by reading from and writing to the Excel file. The answers gained from the FPI approach tend to be more reliable, however, except in those cases where the solution is unstable. Using relaxation may help improve the FPI results. Further information on creating executing both of these models can be found in the User's Guide in Appendix A.

2.3.4 ModelCenter ALS

The ModelCenter ALS was created with the same philosophy as the Model Engineer ALS; the largest difference is that all control of input and output can take place from within ModelCenter rather than an external Excel file. Instead of creating a Fixed Point Iteration model, Optimizer-Based Decomposition could be used instead to enforce constraints between the hydrogen inputs and outputs between heat exchangers. Figures12 and 13 show the feedback loop and OBD approach, respectively. The primary optimization problem is as follows:

Minimize: $W_{sys}=W_{p1}+W_{p2}+W_{cond}+W_{chill}+W_{s1}+W_{s2}$

Subject to:

$$\left(\frac{T_{p1in}-T_{p2out}}{T_{p2out}}\right)^2 \leq 0$$

$$\left(\frac{T_{p2in}-T_{chillout}}{T_{chillout}}\right)^2 \leq 0$$

$$\left(\frac{T_{chillin}-T_{condout}}{T_{condout}}\right)^2 \leq 0$$

$$\left(\frac{P_{p1in}-P_{p2out}}{P_{p2out}}\right)^2 \leq 0$$

$$\left(\frac{P_{p2in}-P_{chillout}}{P_{chillout}}\right)^2 \leq 0$$

$$\left(\frac{P_{chillin}-P_{condout}}{P_{condout}}\right)^2 \leq 0$$

$$\frac{T_{air}-T_{req}}{T_{air}} \leq 0$$

$$\dot{m}_{LH2cond}-\dot{m}_{LH2chill} \geq 0$$

$$\dot{m}_{LH2chill}-\dot{m}_{LH2p2} \geq 0$$

$$\dot{m}_{LH2p2}-\dot{m}_{LH2p1} \geq 0$$

$$T_{LH2p2}-\dot{m}_{LH2p1} \geq 0$$

By Changing:

$$\dot{m}_{LH2p1}, \dot{m}_{LH2p2}, \dot{m}_{LH2chill}, \dot{m}_{LH2cond}, \dot{m}_{EG}$$

$$e_{LH2p1}, e_{LH2p2}, e_{LH2chill}, e_{LH2cond}$$

$$P_{chillin}, P_{p2in}, P_{p1in}$$

$$T_{chillin}, T_{p2in}, T_{p1in}$$

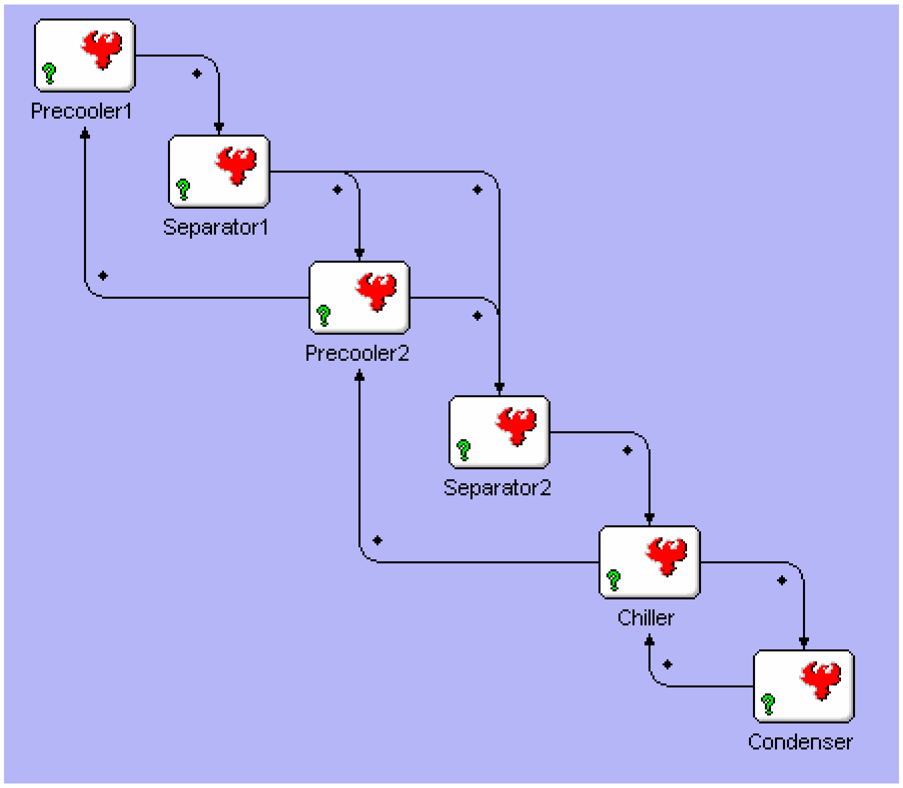


Figure 12: Feedback Version of ModelCenter ALS.

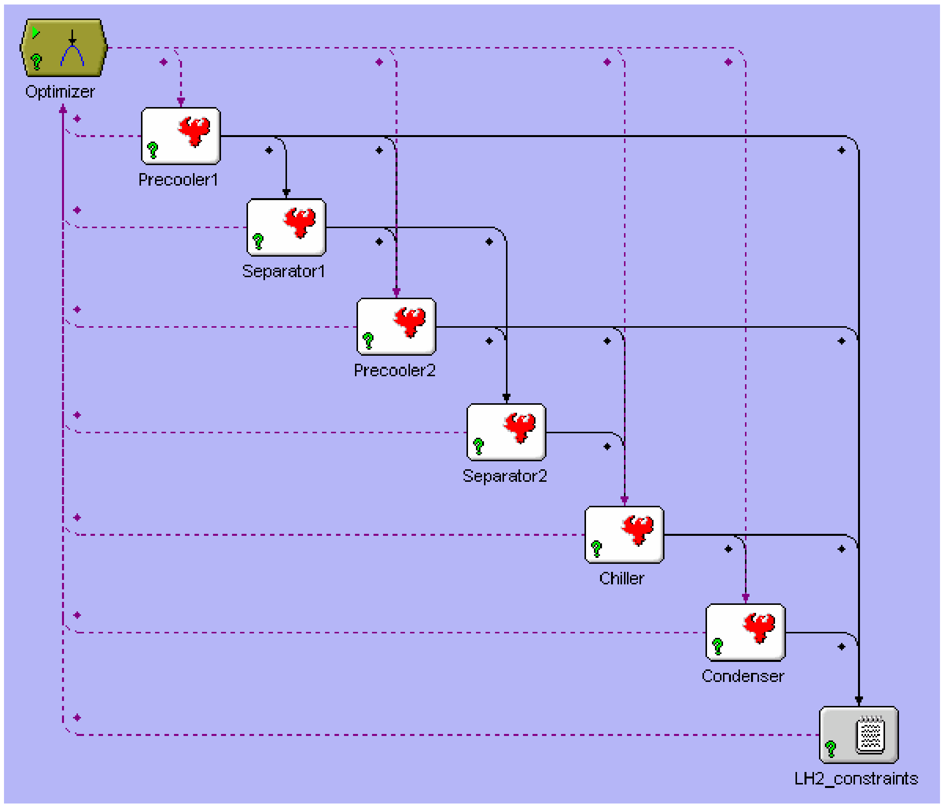


Figure 13: OBD Version of ModelCenter ALS.

The greatest benefit of the ModelCenter ALS is the ability to add an optimizer and scripts to help control execution of the model. For the basic OBD version of the model, the optimizer is designed to alter the cold side mass flows, temperatures, and pressures of each heat exchanger, as well as the hot side effectiveness, in order to achieve a minimum overall system weight. The hot side temperatures and pressures are fed forward through links. A script calculates constraints for the model. These constraints include matching the temperatures and pressures between each heat exchanger's cold sides. This script also enforces the constraint that the air exiting the ALS is at the desired temperature and pressure. These desired values are input by accessing the LH2_constraints script from the Model window.

This is a large and unwieldy optimization for ModelCenter to perform. When executed, it is easy for the optimizer to become bogged down in a certain part of the design space and either stop at an answer that is not a true minimum or cause a crash due to attempts to set variables at inappropriate points. In order to avoid this, some preconditioning can be done to start the model at a reasonable point in the design space. By doing this, the optimizer will be able to find a true minimum rather than exploring undesirable regions of the design space. Through multiple executions, it was found that the Method of Feasible Directions provided the best optimization behavior and results.

A better way of accomplishing this optimization is to take control of some of the design variables away from the optimizer. This also reduces the number of compatibility constraints needed in the OBD. This makes the optimizer much more efficient overall, but makes it even more dependent on the user input.

2.4 ModelCenter vs. Model Engineer

Although the two ALS models rely on the same relationships and equations, the operating environments under which they function significantly affect the relative advantages and disadvantages of each approach. As already explored, ModelCenter's capacity for optimizing and converging the ALS model exceeds that of Model Engineer. The utilities that provide that capability could probably be achieved in Model Engineer, but it would require coding that capability from the ground up. It is much more advantageous to use the already formulated optimizers and parametric study tools within ModelCenter.

However, there are advantages to Model Engineer execution. Creating components is extremely simple in Model Engineer. Whereas for the C++ programs significant thought had to be put into defining algorithms, variable and function declarations, and debugging/compiling, the Object Engineer utility allowed for the "grunt" work of programming to be removed almost entirely. The only coding necessary was defining the variables within the Excel file and then filling in the functions within the Visual Basic class file.

However, the C++ approach had its own merits. The initial setup was more difficult than the Visual Basic approach, certainly. Even so, once the initial programs were written it was very easy to make changes, especially because C++ is such an object oriented language. Any time a variable or function needed to be added to the Model Engineer components, the project file had to be rebuilt and all of the functions redefined. This generally involved a lot of cutting and pasting, which while not difficult is very time consuming and can be frustrating. Additionally, there were some problems between different versions of Model Engineer that caused compiler issues.

The key advantage of the Model Engineer approach is that it promotes the building of libraries of components, from many different authors. While there is nothing stopping ModelCenter from taking a similar approach, Modelogics is dedicated to spreading components created with Object Engineer to all its users. Although it was desired to create the components of the ALS model fresh, preexisting components were available. The ALS could even be incorporated into a larger system model by adding additional engine or aircraft components. For a system that requires little optimization complexity, Model Engineer is an ideal approach to solving the problem of model construction; otherwise ModelCenter remains the superior approach.

3 Validation

Before analysis on the models could begin, it was necessary to validate both the individual components and their ability to model a total system. Data on preexisting systems is in very short supply, simply because these systems barely exist outside of studies done in the 1960s. However, those studies can shed some light on the validity of the ALS model. Model Engineer itself can help verify the models as it had its own heat exchanger and water separator components. Additionally, there has been some recent work on an air liquefaction cycle engine; the Japanese have been working on the LACE ATREX engine for some time and results from that project can be used to validate the heat exchangers.

3.1 Heat Exchanger Validation

Because it was decided to create the ALS model components completely new and not use the already available Model Engineer components, these preexisting components make excellent bases for comparison of heat exchanger model performance. Although the Model Engineer heat exchangers require slightly different inputs for some variables, the ALS is close enough that an easy comparison could be made. For the purposes of validation, the CounterFlowHX component in Model Engineer's Thermal Systems library was used.

The CounterFlowHX component can be configured to force a particular heat exchanger effectiveness just as the ALSHEX component does. Therefore, the hot side output temperatures will always match up and there is no real validation possible (or needed) for this number. However, the cold side properties can be matched. For a cold side input temperature of 200 R, and a range of hot side inputs from 250 to 500 R, the cold side output temperature was found at two heat exchanger effectiveness values. This analysis is summarized in Figure 14. The results are very close, especially when the two input temperatures are close in value. As the difference between hot and cold side inputs increase, however, the ALSHEX component begins to deviate more from the Model Engineer heat exchanger. This difference does not exceed 2% within the maximum range studied; temperature differences of more than 300 R will probably not occur in any single component. The reason for this deviation is probably due to the necessity of estimating specific heat at an average temperature with curve fits. Model Engineer components rely on table lookups from C_p data, and thus probably have slightly more accurate calculations. Nevertheless, the temperature predicting capability of the ALSHEX component would seem more than adequate.

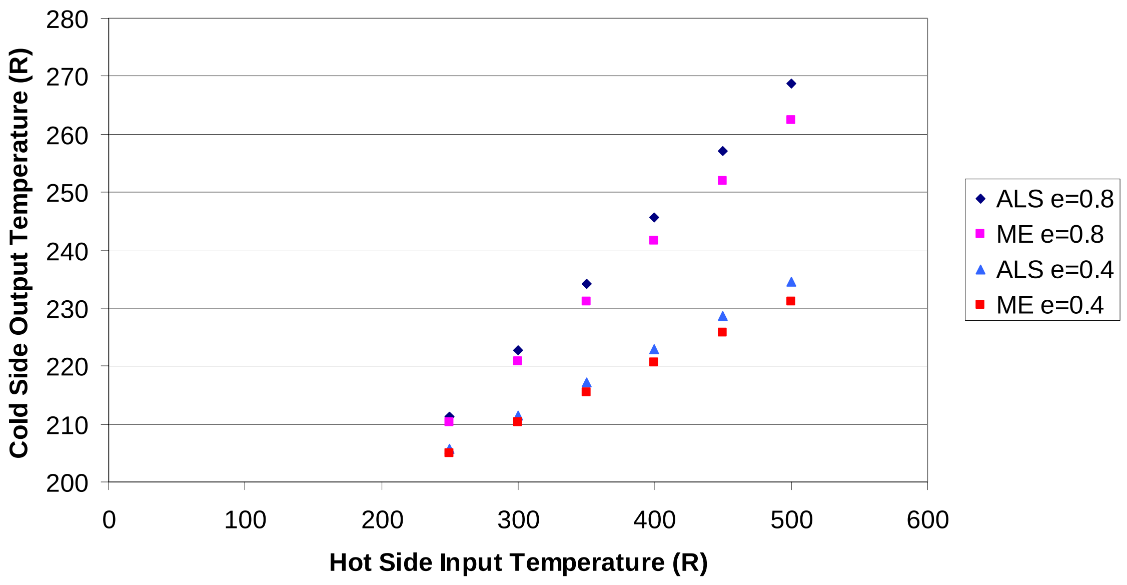


Figure 14: Comparison of Predicted Cold Side Output Temperatures for ME and ALS.

In the case of pressure drop calculations, it would appear that CounterFlowHX relies on the same relationships as those in the ALSHEX component (Ref 9, 122). The pressure drop across the heat exchangers always matched up for any input value.

The weight and sizing results Model Engineer provided did not prove to match as well, unfortunately. The graph in Figure 15 shows the percent discrepancy between the ALS heat exchanger and Model Engineer's CounterFlowHX. The discrepancies for the volume are decent, although they do reach a maximum of nearly 5%. More concerning is the high difference between the weight estimates, up to 25% for the highest temperature difference. Again, the percent error rises as the difference between hot and cold side temperatures rises, pointing to a possible issue with specific heats. The volume error does not increase in precisely this manner; although it appears to be fluctuating, the errors displayed in the graph are absolute errors. They increase from almost -5% at the smallest temperature difference to almost 5% at the largest. This behavior cannot simply be attributed to the C_p calculation, although it remains to be discovered why this is so.

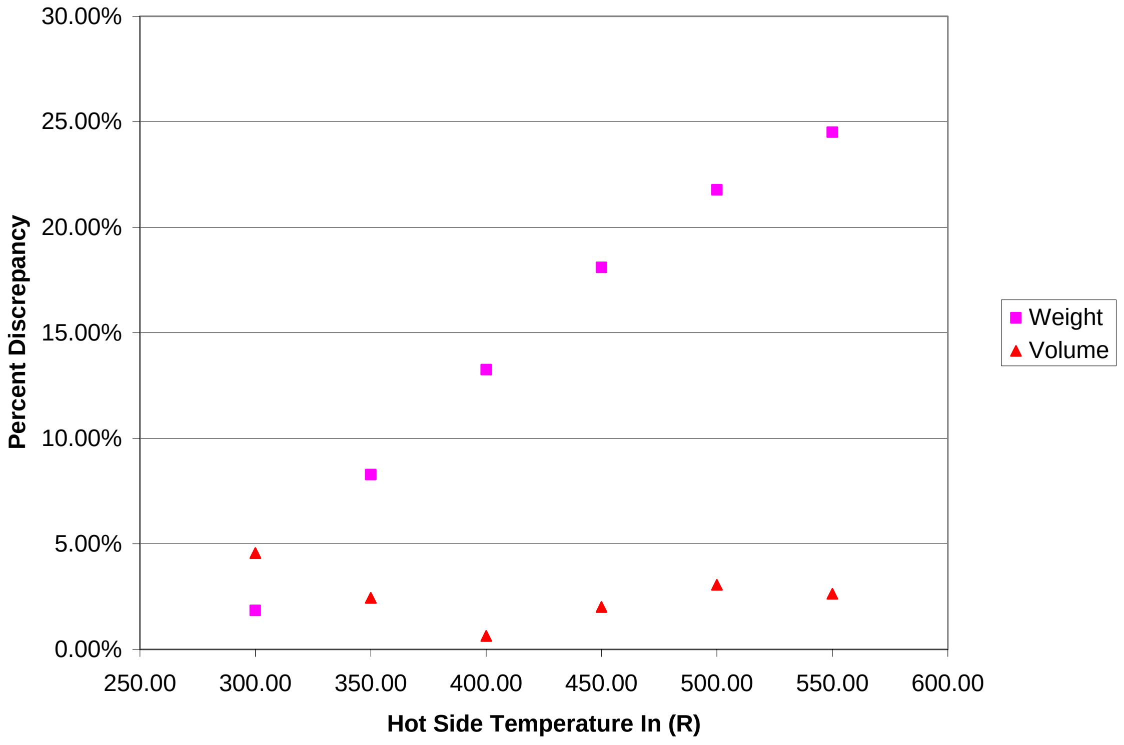


Figure 15: Absolute Percent Discrepancy for Weights and Sizing.

Additional performance validation could be achieved by comparing results of the ALS heat exchanger component with the ATREX engine under development in Japan. They conducted an experiment testing some precooler designs running air and hydrogen. Data available for two of these engines, the ATREX8-3 and the ATREX8-5, are available in Table 2, along with the predicted performance by the ALS model given the same inputs (Ref 10). The model predicted the performance of the ATREX8-3 precooler very well. The greatest error occurred in the pressure of the outgoing hydrogen, 2.25%. The outgoing hydrogen temperature had an error of 1.5%. These errors are somewhat significant, but overall are acceptable from a conceptual design standpoint. The ATREX8-5 predictions, however, were somewhat more troubling. The pressure predictions were very good, both within 1% error. However, the prediction for the outgoing hydrogen temperature was off by almost 10%. This is a significant error, although it is difficult to speculate why it exists as there is almost no other information about the precooler setup available. There may be factors unaccounted for in the model, or there may be something about the precooler itself that makes the model inappropriate. Overall, the ability of the heat exchanger components to correctly model a real system seems fairly good.

Table 2: ATREX Precooler Data and ALS Predicted Performance.

Fluid	Property	Actual	ALS	Percent Error	Actual	ALS	Percent Error
	Engine	ATREX8-3		--	ATREX8-5		
Hydrogen	Mass Flow In (lb/min)	32.94		--	38.50		
	Temperature In (deg R)	55.80		--	55.80		
	Temperature Out (deg R)	394.20	400.10	-1.50%	365.40	397.43	-8.77%
	Pressure In (psia)	368.40		--	532.30		
	Pressure Out (psia)	342.29	349.98	-2.25%	506.19	505.68	0.10%
Air	Mass Flow In (lb/min)	959.18		--	992.25		
	Temperature In (deg R)	502.20		--	504.00		
	Temperature Out (deg R)	338.40	338.37	0.01%	320.40	320.24	0.05%
	Pressure In (psia)	14.77		--	14.63		
	Pressure Out (psia)	13.98	14.03	-0.32%	13.79	13.90	-0.80%
	Effectiveness	0.37		--	0.41		

3.2 Water Separator Validation

The water separator component can be validated from two different sources. Subscale hardware was built and tested by Marquardt and Garrett AiResearch in the 1960s, and data was obtained about how much water was removed from an operating water separator as a plane flew through the atmosphere. In Figure 16, the water removed per mass flow rate of air entering the separator can be seen for an initial relative humidity of 70%. This test had a duration of 80 seconds. To compare this data to the ALS separator component, the water removed from the system per minute was multiplied by this flight time and divided by the mass flow of air. As seen in the figure, the results matched very closely with that of the test system. This system had no ethylene glycol added, nor is any data available which would suggest the performance of an ethylene glycol water separator.

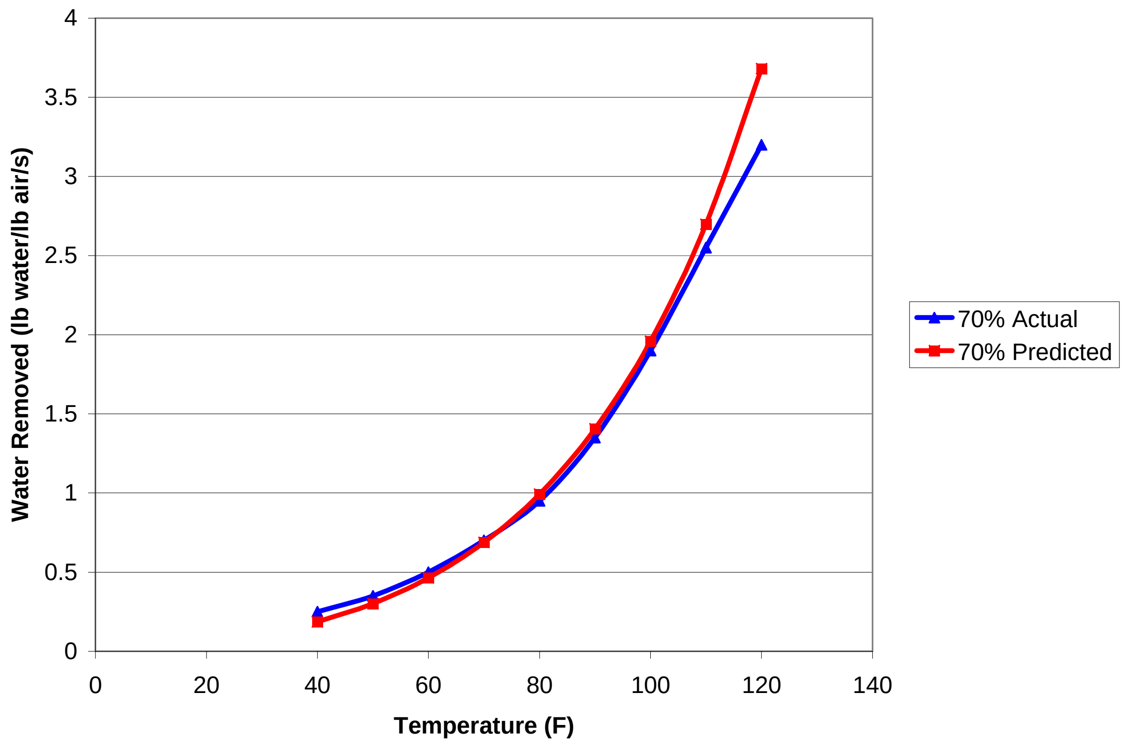


Figure 16: Water Removed vs Inlet Temperature.

Model Engineer's separator component can also be used to validate some of the data, although it also has no capability to add ethylene glycol to the mixture so it is impossible to fully validate the water separator data. The two models were executed for a reference relative humidity of 70% at a range of reference temperatures from 510 to 580 R, and an actual air temperature of 500 R. The two models show almost perfect agreement, especially when the reference temperature is close to the actual temperature. At the far end of the graph in Figure 17, the percent difference has risen to 5%, which while not optimal is still reasonable for a conceptual design tool.

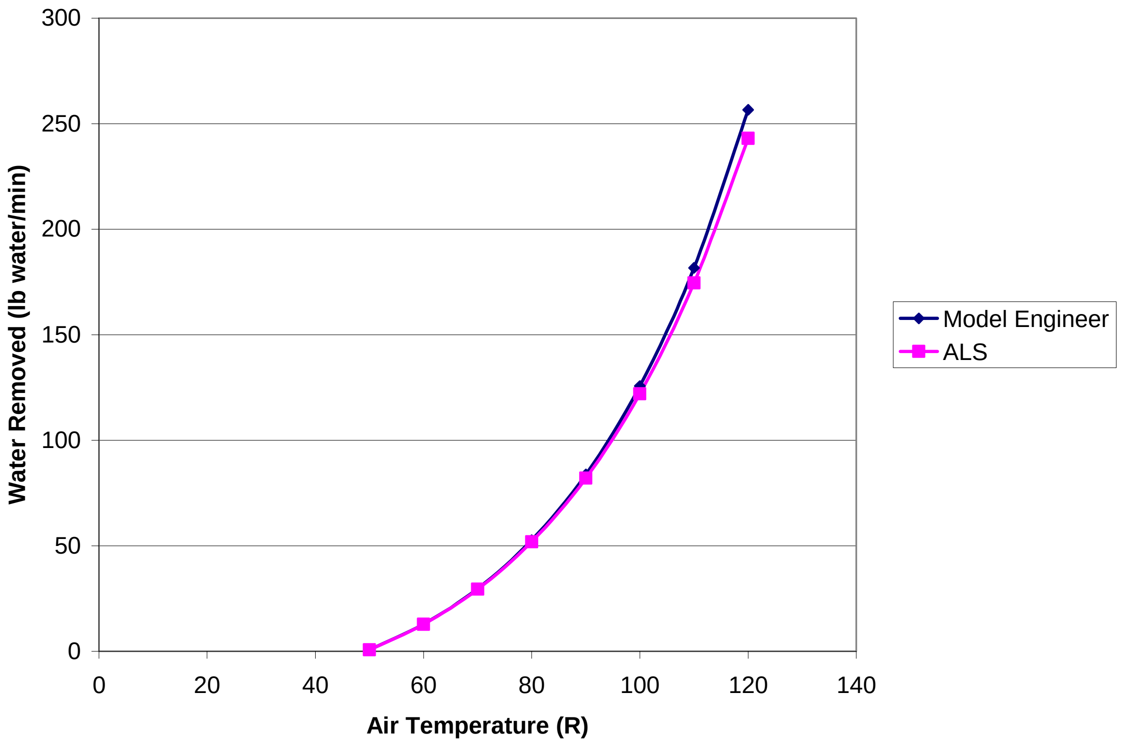


Figure 17: Comparison of ME and ALS Water Removal.

Given that data concerning the addition of ethylene glycol to a fast moving air stream is unavailable, it was impossible to verify the model's performance for that variable. However, if the assumption that the ethylene glycol is completely and evenly absorbed by the water is a good one, then the component should be very accurate as it is known that Raoult's law is very accurate for non-electrolytic solutions.

4 ALS Model Analysis

With the model validated and running correctly, the original goal of the project could be reached. The model was mandated to run at sea level static (SLS) conditions, but an additional analysis was conducted to test the model at other points in the design space, and to gain an understanding of what drives the model. The program SCCREAM was used to simulate the flight of an aircraft through the atmosphere at high speeds and altitudes. The inlet conditions were then extracted from SCCREAM's output deck, and fed to the ALS model to determine system efficiency at those conditions.

4.1 Sea Level Static

The conditions prescribed by Larry Hunt of SAIC for SLS conditions are listed in Table 3. These conditions represent an aircraft taking off, most likely under power of a rocket engine as the ALS model is designed primarily to act as the front end to a LACE RBCC engine. There are no specific requirements for water removal, but if the component works properly 97.75% of the water should be removed from the air stream if both separators work given a separator efficiency of 85%. Additionally, the system efficiency of pounds of liquid air produced per pound of hydrogen necessary to produce it should be around 4 to 5.

Table 3: SLS Air Liquefaction System Parameters.

Fluid	Property	Input Variables	Output Requirements
Air	Mass Flow	6000 lbm/min	~6000 lbm/min
	Temperature	540 deg R	~130 R
	Pressure	14.7 psia	~10 psia
	Velocity	100 ft/sec	
	Relative Humidity	80%	
LH2	Temperature	40 R	
	Pressure	800 psia	

Using the complete Optimizer Based Decomposition approach where all compatibility constraints are handled by the optimizer, it took 5.5 minutes to converge to a solution. The large number of constraints tended to bog the optimizer down and initially sent it in directions that were far off minimum to meet constraints, as can be seen in Figure 18. The most important results from the optimized ALS are summarized in Table 4. The limiting components seem to be the water separators and the condenser. In the case of the condenser, this makes sense as it has to do much of the work of creating the liquid air. However, due to the fact that the water separator weight functions were not able to be validated, those numbers may be off.

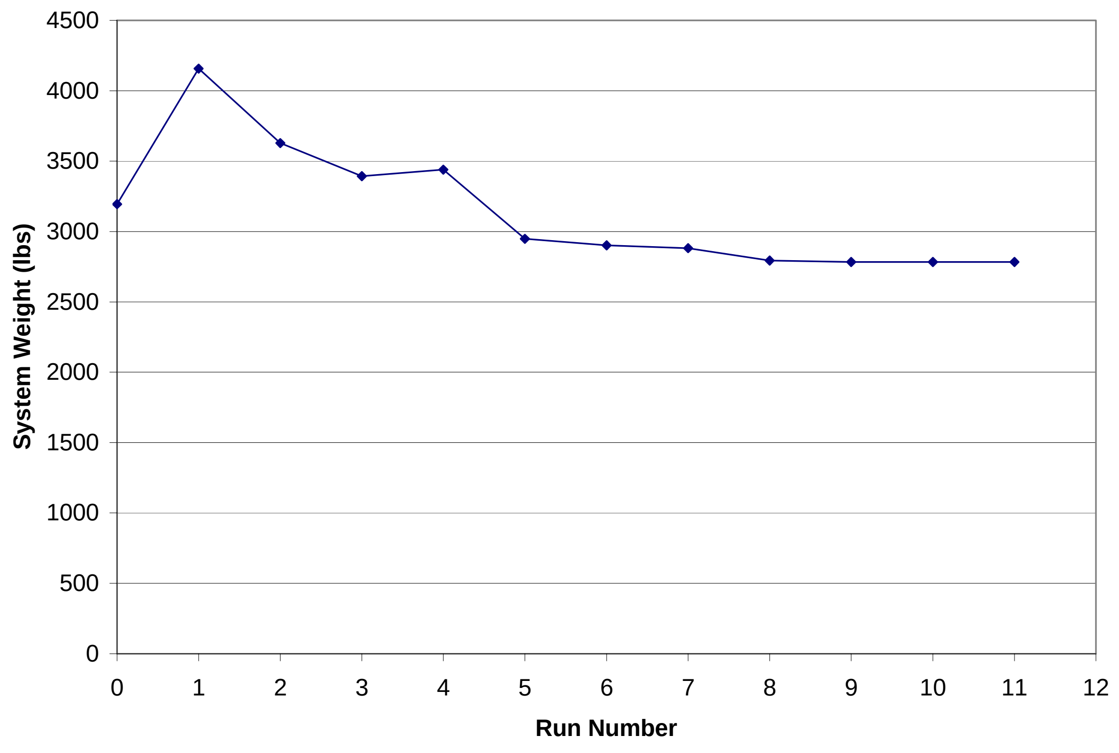


Figure 18: Optimization History of ALS Model for SLS Conditions.

In any event, the system has met the design goals. The mass flow of liquid air produced is 5935 lb/min at a temperature of 130 R and a pressure of 10.81, which roughly match the goals set in Table 3. Furthermore, the water separators worked correctly and removed most of the water from the air stream which should guarantee that the engine will continue to run. The highest mass flow of hydrogen necessary to run the system is 1419 lb/min, which results in a system efficiency of 4.18 lb liquid air per lb liquid hydrogen, within the stated requirements.

Table 4: Results of Optimizer Based Decomposition on SLS ALS.

Fluid	Property	Precooler1		Separator1		Precooler2		Separator2		Chiller		Condenser	
		In	Out	In	Out	In	Out	In	Out	In	Out	In	Out
Air	**T (R)**	540.00	513.99	513.99	513.99	513.99	489.53	489.53	489.53	489.53	382.84	382.84	129.67
	P (psia)	14.70	13.97	13.97	13.27	13.27	12.60	12.60	11.97	11.97	11.37	11.37	10.81
	Mdot (lbm/min)	6000.00	6000.00	6000.00	5957.73	5967.88		5967.88	5935.21	5935.21	5935.21	5935.21	
	e	0.10				0.10				0.41		0.74	
H2	**T (R)**	279.92	294.87			269.36	279.92			229.01	267.72	40.00	231.04
	P (psia)	686.91	652.57			721.98	685.88			759.22	721.26	800.00	760.00
	Mdot (lbm/min)	750.01				992.03				1173.36		1419.48	
	Weight (lbm)	43.81		561.60		42.95		558.59		187.53		1388.98	
	Volume (ft3)	0.62		25.10		0.60		26.97		3.20		31.00	

4.2 Additional Cases

Finally, it was desired to attempt to use the model at different conditions other than SLS. Using the program SCCREAM most of the data for the air entering the inlet at speeds above Mach 2.5 and altitudes above 30000 ft could be obtained. However, the humidity at these altitudes is unknown, so the water separator components were removed for this analysis. The optimization problem had to change somewhat. The optimization problem for this high speed case is the same as for the SLS system, but the design variable for the mass of ethylene glycol added to the system (m_{EG}) was removed as the water separators are not being used.

The results of one optimization, which finished in roughly 6 minutes, are shown in Table 5 and Figure 19 below. The input conditions are for the entrance to a Scramjet inlet at Mach 2.5 at an altitude of 40000 ft. The exact model used for the SLS case had trouble converging and used zero gradients. It became clear that the system is very sensitive to the mass flow of hydrogen supplied to the condenser. Enough mass flow of hydrogen must be specified, or the optimizer will not be able to find a feasible and viable solution. However, a solution was found; the resulting system has a system efficiency of 5.24. Clearly the SLS system is easier to operate at the desired efficiency, but this result is very close to the desired goal. A total of 15765 lb air/min in liquid form at a pressure of 47.63 psia is the result of this system.

It is interesting to note that in both the SLS and high speed cases, the condenser is a large component. This is due primarily to the fact that it experiences the largest mass flow rates; the other components can use lower hydrogen mass flow rates, but hydrogen flow can never be increased once it has been reduced. Another behavior of the ALS system that becomes more obvious for the high speed case is the optimizer dependence on initial values for mass flow rates. The optimizer explores a large range of heat exchanger effectiveness values, but rarely strays far from the initial guesses for mass flow rate. Therefore, it is vitally important to make good guesses for these flow rates if a solution near the system's true minimum is to be found.

Table 5: Results of Optimizer Based Decomposition on High Speed and Altitude ALS.

Fluid	Property	Precooler1		Separator1		Precooler2		Separator1		Chiller		Condenser	
		In	Out	In	Out	In	Out	In	Out	In	Out	In	Out
Air	T (R)	882.00	840.42			840.42	650.95			650.95	422.62	422.62	129.67
	P (psia)	58.48	55.56			55.56	52.78			52.78	50.14	50.14	47.63
	Mdot (lbm/min)	15765.00	15765.00			15765.00	15765.00			15765.00	15765.00	15765.00	15765.00
	e	0.10				0.42				0.64		0.77	
H2	T (R)	466.15	881.99			386.74	464.11			295.78	385.07	40.00	295.57
	P (psia)	685.90	651.61			722.00	685.90			760.00	722.00	800.00	760.00
	Mdot (lbm/min)	2700.00				2798.82				2898.38		2999.83	
	Weight (lbm)	111.76				545.82				903.12		3884.94	
	Volume (ft3)	1.78				10.75				19.03		99.48	

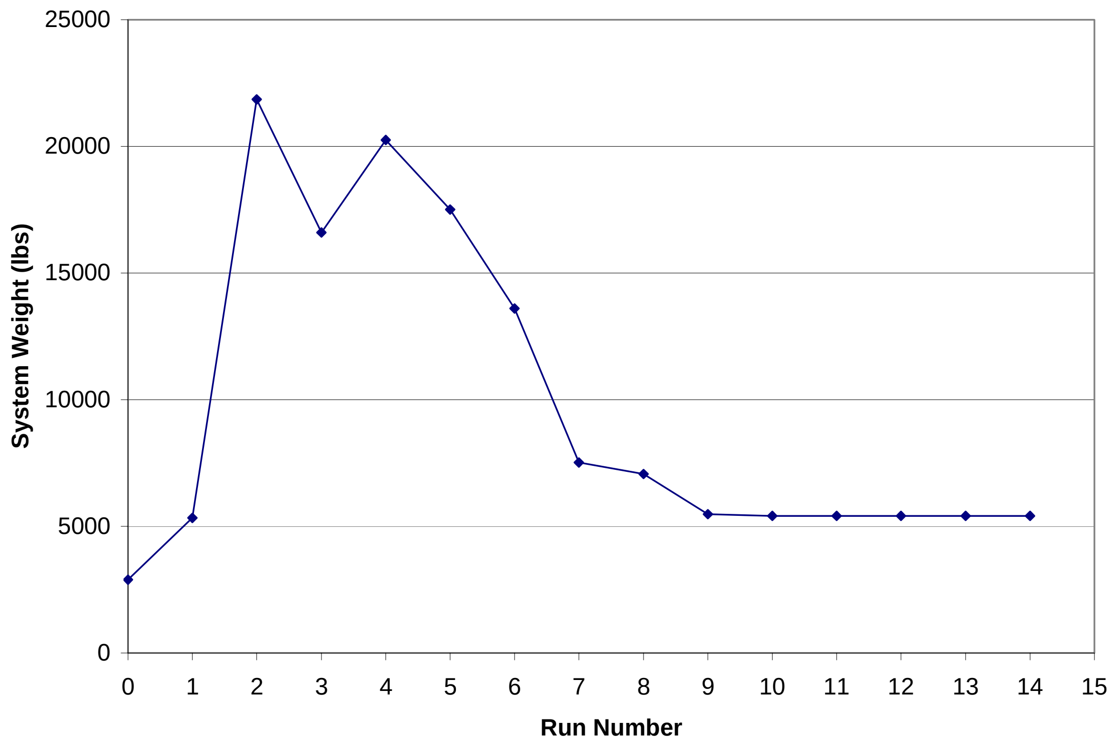


Figure 19: Run History for High Speed Case (Mach = 2.5).

4.3 Trade Studies

Finally, some trade studies using the model were conducted. The use of the para/ortho catalyst was investigated for a single heat exchanger, as it can only be used when the hydrogen is in liquid form; this condition will only occur in the condenser. Although the results of the original study (Ref 3-3) showed that the desired catalyst performance was unattainable, the actual catalyst weight needed to convert hydrogen to its ortho form is sufficient to increase performance without gaining weight. The data suggests that at higher mass flows of hydrogen, this will no longer be the case; however, by the time this occurs more hydrogen would be used than liquid air produced. The previous system weight optimization showed that such high system efficiencies are not necessary. Figure 20 demonstrates this behavior.

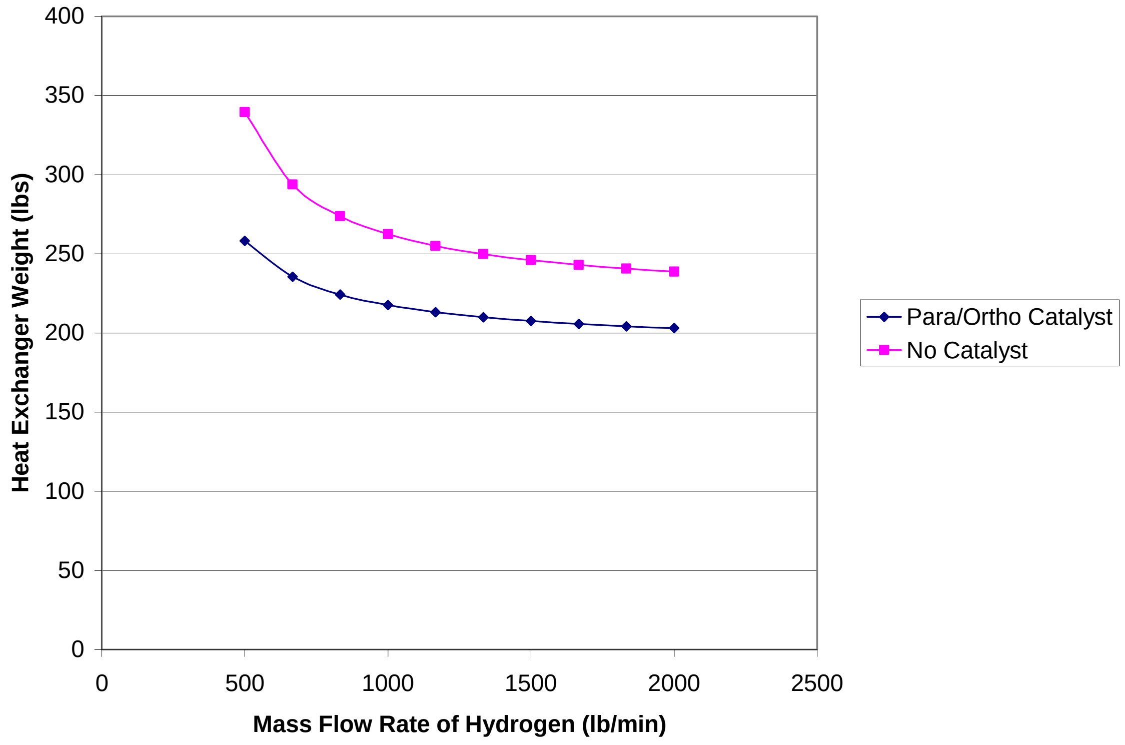


Figure 20: Effect of Para/Ortho Conversion Catalyst on Heat Exchanger Weight.

Additionally, a study was performed to determine how the required system efficiency affects system weight. The ALS goal was to obtain system efficiencies of 4 to 5; in other words, 4 to 5 times the amount of liquid air is created for every unit of liquid hydrogen required. Obtaining higher system efficiencies may be possible, but lower system efficiencies will result in lower weight systems. Figure 21 shows the behavior of the system weight as the system efficiency is raised. The behavior is fairly linear for low efficiency values, but rapidly begins to become exponential after an efficiency of 3. A system efficiency of 6 was not even possible to obtain (or at least could not be found by the optimizer). A system efficiency of 5 seems to be roughly the best that could be achieved.

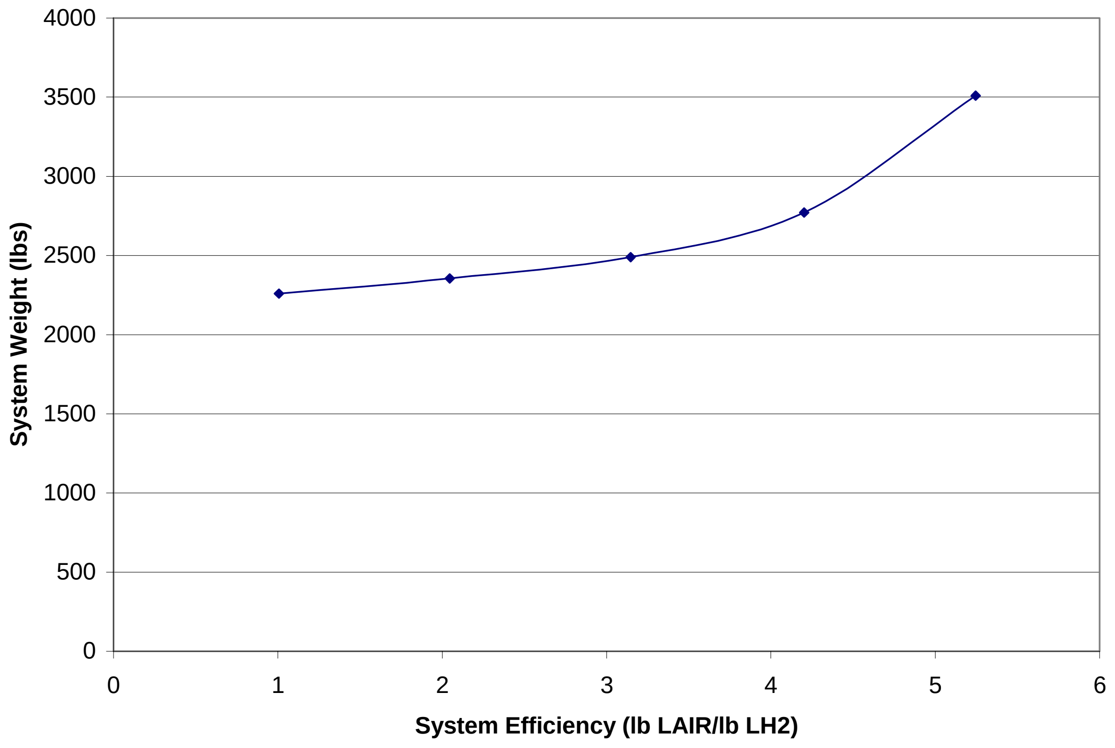


Figure 21: System Weight vs. Required System Efficiency.

Conclusions

Air liquefaction has not become prevalent as a technology for enabling spaceplanes. This is in large part due to the design challenges and uncertainty in creating an engine cycle that can incorporate the heavy heat exchangers necessary to cool the air to its liquid state. Additionally, ice water fouling can ruin the effectiveness of such schemes to the point where the engine fails. In order to better understand the process of air liquefaction and to assess its suitability for incorporation into spaceplanes, the ALS tool has been created to model any air liquefaction scheme.

The ALS tool is comprised of two components: heat exchangers and water separators. These components have been validated against components already created for Modelogics' Model Engineer software. Additionally, the heat exchanger performance was validated against the ATREX engine currently in development. Overall, the components performed as expected, except in the category of weights and sizing. The heat exchanger components match fairly well with the Model Engineer components, although errors get worse as the temperature difference between the two working fluids rises. The water separator component was not validated for weights and sizing due to a dearth of data.

These components can be connected in virtually any configuration to demonstrate air liquefaction potential. In the scope of this project, a particular ALS design as specified by Larry Hunt of SAIC was explored. This ALS contains four heat exchangers and two water separators, with the capability to add an ethylene glycol spray to the air flow to prevent water fouling and improve water separation performance.

The resulting model can be executed in two design frameworks: Modelogics' Model Engineer and Phoenix Integration's ModelCenter. While both of these frameworks offer advantages to a user, it is within ModelCenter that the ALS really reaches its potential. The inability of Model Engineer to conduct complex optimizations leaves it suitable only for examining the performance of point designs. Within ModelCenter, the ALS model can be optimized upon until a minimum system weight is achieved and all performance requirements are met.

This optimization was conducted for a specific set of conditions. SAIC provided the author with sea level static conditions after the engine inlet. These conditions and design requirements were listed in Table 3. This optimization completed successfully, and in only 5.5 minutes. The resulting ALS weighs 2783 lbm and produces 5935 lbm/min of liquid air at 129 R and 10.8 psia. This ALS requires a mass flow of liquid hydrogen of 1419 lb/min, for a total system efficiency of 4.18.

The ALS model was also executed for conditions at Mach 2.5 and 40000 ft. The results here were not as conclusive due to the lack of data of humidity levels at high altitudes; the high speed ALS does not include water separators. The limited optimization completed, however, resulting in a system producing 15675 lb/min of liquid air given almost 3000 lb/min of liquid hydrogen. This ALS weighs 5350 lbm and has an overall system efficiency of 5.225. This does not quite reach the goals set forth for the SLS system, but is nevertheless an encouraging result.

Despite some discrepancies in the validation of the weights and sizing parts of the ALS components, the performance aspects of all ALS components work admirably at low atmospheric conditions. Despite the issue of weight error, the weights and sizing variables do behave in a logical manner (rising and falling when they should), which allows the system to be optimized, and for that optimization to make sense. Furthermore, despite the complex nature of the design problem (15 design variables and 13 constraints) an optimum was reached in a fairly quick amount of time.

The ALS model is therefore a useful tool for evaluating the performance of an air liquefaction engine cycle at low atmospheric conditions. Its applicability to high speed flight at higher altitudes is yet undetermined due to lack of concrete data. Given the inputs behind the inlet of an engine, it will determine the mass flow of hydrogen, component weights, and water fouling measures necessary to produce liquid air for use in the combustion chamber. Given its modularity, it could easily be incorporated to a more complex system model within Model Engineer or ModelCenter to model an entire engine or even the whole spacecraft of which that engine is a part.

References

1. E., Mark "Weather Programs & Algorithms: Calculating Humidity Properties", http://snowball.frogspace.net/js/wxalgs2.html, May 02, 1999.

2. Escher, William J.D., "Cryogenic Hyrdogen-Indcued Air Liquefaction Technologies For Combined-Cycle Propulsion Applications," European RBCC Workshop, Delft, The Netherlands, November 6-9, 1995

3. Escher, W.J.D., Doughty, D.L., "Assessment of Cryogenic Hydrogen-Induced Air Liquefaction Technologies," Astronautics Technology Center, Astronautics Corporation of America, September 1986.

4. Hodge, E. "Model Engineer User's Guide," Modelogics, Inc.

5. "Raoult's Law," http://en.wikipedia.org/wiki/Raoult's_law, June, 2003.

6. Sherif, S.A., Sullivan, N., Ihas, G., Zhou, D., "UF Low Temperature/Hydrogen Group Task #3 Hydrogen Storage," http://www.fsec.ucf.edu/hydrogen/pdf-slides-01-2003/uf-t3b.pdf.

7. "Technical FAQs" http://www.ashchem.com/adc/chemicals/faq_answer.asp?typeID=3&is_header=N, 2004.

8. "This file contains my notes on psychrometrics," http://courses.ncsu.edu/classes/wps203001/online/psychrometrics.htm.

9. Whitney, A.E., Whitman, C.E., Li, K.C., "Development of Integrated Environmental Control System Designs for Aircraft," Vol. III., McDonnell Aircraft Company, McDonnel Douglas Corporation, St. Loius, MO, May, 1972.

10. Harada, K., Yamauchi, H., Tanatsugu, N., Sato, T., Okabe, Y., Hamabe, K., Tomike, J., Kazari, M., "Development Study on Precooler for Atrex Engine," 1997.

11. Issacci, F., Farr, J.L., Jr., Wassel, A.T., Griethuysen, V.V., "An Integrated Thermal Management Analysis Tool," 1996 JANNAF Propulsion Meeting, Colombia, MD, December, 1996.

Appendix A--ALS Model User Guide

The Air Liquefaction System model is designed to allow the user to simulate the operation of the front end of an air liquefaction engine cycle. It consists of two types of components; heat exchangers and water separators. Although the model by default simulates a system with four heat exchangers and two water separators, the components can be used in any combination to model different air liquefaction schemes, or even simpler air/hydrogen heat exchange systems.

The model is available for execution in two operating environments; Modelogics Inc.'s Model Engineer, and Phoenix Integration's ModelCenter. Additionally, the components that comprise the model may be executed as stand alone C++ compiled executables.

The purpose of this guide is to educate new users in the use of this model and its components. It discusses in detail the input and output for both types of execution (Visual Basic and C++), and the process of combining components into a system. Therefore this guide is separated into three primary sections: Heat Exchanger Component, Water Separator Component, and ALS Model.

Heat Exchanger Components in C++

The simplest way to use the heat exchanger component is to run its executable. The component HX.exe was written in C++ and relies on simple file based input and output, and can be executed on any Windows operating system. Source code is available for compiling for different operating systems; it has been successfully compiled with Borland's BCC55 for Windows execution and the Unix compiler g++.

Shown below is the notional input file for the ALSHEX component. The first five lines are merely a header; these headers can be changed as desired as long as the number of lines before the actual data begins remains constant. Both the executable and the ModelCenter fileWrapper depend on knowing where the data are. The input file is fairly self-explanatory, but the entries in it will be delineated here.

Notional Input File

```
Header1

---------------------------------------------------
Header2
---------------------------------------------------
TH
PH
MH
HH
STATEH
EH
VH
```

```
TC
PC
MC
HC
STATEC

EG
TUBED
MAT
```

Table A1 lists all of the variables along with their properties. The first seven input variables describe the hot side (air) inputs. While the lower bounds of most of these variables is 0, entering a value of zero for any of these variables will likely cause a system crash; having a temperature of absolute zero, putting no mass flow through the system, or specifying a 0 effectiveness can cause difficulty. The three integer variables (STATEH, STATEC, and MAT) reflect user choices, not literal numbers. For the state variables, 0 represents that the fluid is a gas, and 1 represents a liquid. For material choices, there are currently only two choices, aluminum (1) and stainless steel (2). A sample input file can be found at the end of this section.

Table A1: ALSHEX System Inputs

Variable Name	Group	Description	Units	Type	Lower Bound	Upper Bound
TH	Hot Side	Temperature in	deg R	Double	0	--
PH	Hot Side	Pressure in	psia	Double	0	--
MH	Hot Side	Mass Flow in	lb/min	Double	0	--
HH	Hot Side	Enthalpy in	Btu/lb	Double	--	--
STATEH	Hot Side	State in	Gas, Liquid	Integer	0	1
EH	Hot Side	Heat Exchanger Effectiveness	--	Double	0	1
VH	Hot Side	Flow Velocity In	ft/sec	Double	0	--
TC	Cold Side	Temperature in	deg R	Double	0	--
PC	Cold Side	Pressure in	psia	Double	0	--
MC	Cold Side	Mass Flow in	lb/min	Double	0	--
HC	Cold Side	Enthalpy in	Btu/lb	Double	--	--
STATEC	Cold Side	State in	Gas, Liquid	Integer	0	1
EG	Sizing	Mass Flow of Ethylene Glycol	lb/min	Double	0	--
TUBED	Sizing	Tube Diameter	ft	Double	0	--
MAT	Sizing	Material Choice	Aluminum, Stainless Steel	Integer	1	2

The executable is run using the command HX.exe (or ALSHEX if it has been compiled using the g++ command line g++ hx.cpp –o ALSHEX). This will produce the output file. If using the ModelCenter version of the ALSHEX component, the program will be executed automatically by running the component. The output file has the notional format below:

Notional Output File

Heat Exchanger Output File

```
-------------------------------------------------------
Property              Hot Side       Cold Side
-------------------------------------------------------
                      THOUT          TCOUT
                      PHOUT          PCOUT
                      MHOUT          MCOUT
                      HHOUT          HCOUT
                      CPH            CPC
                      RHOH           RHOC
                      STATEH         STATEC


                      EHOUT          ECOUT
                      V
                      A
                      L
                      TUBES
                      CU
                      RI
                      WTCORE
                      WTHEX
                      ERRFLAG
```

Again, the properties of these variables are listed below in Table A2. Many of the output variables are similar to the input variables, but there are some differences. The specific heat and density of the fluids at the exits are output just to provide more information on the outgoing flow. Additionally, the actual effectiveness of the heat exchanger is output; these numbers may not be the same as what was indicated in the input file if hydrogen mass flow is insufficient to achieve that effectiveness. The variables CU and RI represent cost units and reliability index respectively, but the relationships providing these values did not provide units; therefore, they should only be used to gauge relative prices and reliabilities of different heat exchanger designs. ERRFLAG indicates if the mass flow was insufficient to gain the specified performance out of the heat exchanger. A value of 0 indicates no error, while a value of 1 indicates a flow deficiency. A sample output file is shown at the end of the section.

Table A2:ALSHEX System Outputs

Variable Name	Group	Description	Units	Type
THOUT	Hot Side	Temperature out	deg R	Double
PHOUT	Hot Side	Pressure out	psia	Double
MHOUT	Hot Side	Mass flow out	lbm/min	Double
HHOUT	Hot Side	Enthalpy out	Btu/lbm	Double
CPH	Hot Side	Specific Heat out	Btu/lbm/R	Double
RHOH	Hot Side	Density out	lbm/ft3	Double
STATEH	Hot Side	State out	Gas, Liquid	Integer
EHOUT	Hot Side	Effectiveness out	--	Double
TCOUT	Cold Side	Temperature out	deg R	Double
PCOUT	Cold Side	Pressure out	psia	Double
MCOUT	Cold Side	Mass flow out	lbm/min	Double
HCOUT	Cold Side	Enthalpy out	Btu/lbm	Double
CPC	Cold Side	Specific Heat out	Btu/lbm/R	Double
RHOC	Cold Side	Density out	lbm/ft3	Double
STATEC	Cold Side	State out	Gas, Liquid	Integer
ECOUT	Cold Side	Effectiveness out	--	Double
V	Sizing	Heat Exchanger Volume	ft3	Double
A	Sizing	Cross-sectional Area	ft2	Double
L	Sizing	Length	ft	Double
TUBES	Sizing	Number of heat exchanigng tubes	--	Integer
CU	Sizing	Cost Units	?	Double
RI	Sizing	Reliability Index	?	Double
WTCORE	Sizing	Core Weight	lbm	Double
WTHEX	Sizing	Overall Heat Exchanger Weight	lbm	Double
ERRFLAG	Sizing	Error Flag	No error, insufficient mass flow	Integer

Heat Exchanger Operation in ModelCenter

To operate the ALSHEX component in ModelCenter, a fileWrapper was created. Filewrappers are stored on an Analysis Server and tell ModelCenter where to find input and output variables. The fileWrapper at the end of this section details the input, execution, and output of the heat exchanger component. The variables are indexed by row and field number in the input and output files. For more information on the operation of fileWrappers, please consult the ModelCenter documentation.

To load the HX component, simply connect to an Analysis Server that hosts the HX fileWrapper, executables, and input/output files. Drag the component into the project window, and the component will automatically load. Input variables are displayed by icons with green arrows heading into a box, while output variables are displayed by icons with blue arrows heading out of the box. The input and output variables are grouped by the fileWrapper to help organize them. Input groups include the following variables:

HotSideIN: Contains all input applying to hot (air) side inputs

ColdSideIN: Contains all input applying to cold (hydrogen) side inputs
SizingIN: Contains all input applying to sizing inputs

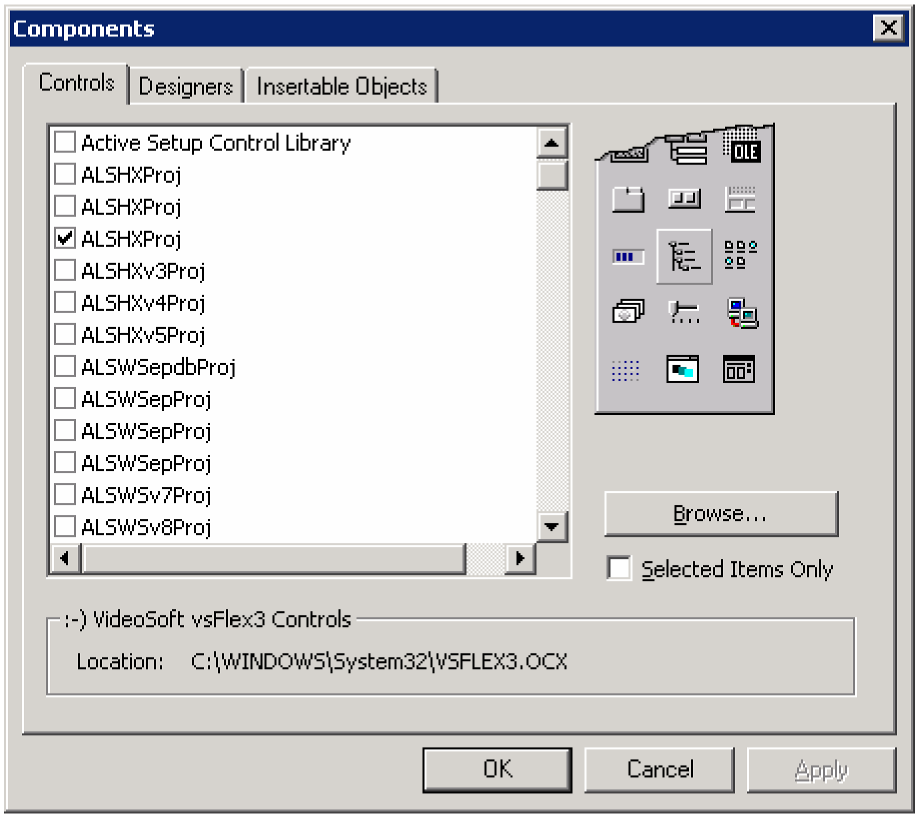
Output groups include the following variables:

HotSideOUT: Contains all outputs pertaining to the hot (air) stream
ColdSideOUT: Contains all outputs pertaining to the cold (hydrogen) stream
SizingOUT: Contains all weight, sizing, cost, reliability, and error outputs

To execute the model, either right click on the component and select Run, or click on the arrow in the upper left hand corner of the component in the Project Window.

Heat Exchanger Operation in Model Engineer

To operate the ALSHX component in Model Engineer, you must have Visual Basic, the Model Engineer libraries, and the ALSHX component installed and registered on your system. Open Visual Basic, and start a new project (Standard EXE). A form will be created. Hit Alt-P or click on "Project" in the Visual Basic Toolbar and hit Ctrl-T or select "Components…" to open the Components window. Scroll through the available components until you find ALSHXProj. Check the box next to the component (see Figure A1) and press OK. A new component will appear on the toolbar to the left. By clicking and dragging this icon to the form, the ALSHX component can be added to the form.



- [ ] Active Setup Control Library
- [ ] ALSHXProj
- [ ] ALSHXProj
- [x] ALSHXProj
- [ ] ALSHXv3Proj
- [ ] ALSHXv4Proj
- [ ] ALSHXv5Proj
- [ ] ALSWSepdbProj
- [ ] ALSWSepProj
- [ ] ALSWSepProj
- [ ] ALSWSepProj
- [ ] ALSWSv7Proj
- [ ] ALSWSv8Proj

- [ ] Selected Items Only

:-) VideoSoft vsFlex3 Controls
Location: C:\WINDOWS\System32\VSFLEX3.OCX

Figure A1: Visual Basic Component Window

There are two ways to edit the inputs and view the outputs of the ALSHX component. First, you may use the Properties window and select the ALSHX component from the drop down menu. The variables of the component will be listed below, and may be edited by clicking in the appropriate box and typing in a new number. Secondly, you can right click on the component itself and select “Edit”. Then right click again and select “Properties” to open the component. There are four property pages; the first simply contains information for Model Engineer. The next three pages contain hot side, cold side, and sizing properties respectively. Any variable boxes colored blue are read only, but any other variable boxes may be edited. To see the results of an input change, simply press the Apply button and the output values will update automatically.

Sample Heat Exchanger Input/Output/FileWrapper Files:

Sample Input File (hx.in)

```
Heat Exchanger Input File

-----------------------------------------------------
Hot Side/ColdSide
-----------------------------------------------------
492.893                 Hot Side Temperature
11.9699                 Hot Side Pressure
5924.34                 Hot Side Mass Flow
-24.0676                Hot Side Enthalpy
0                       Hot Side State
0.8                     Hot Side Effectiveness
100.0                   Hot Side Velocity

180.879                 Cold Side Temperature
798.0                   Cold Side Pressure
1500.0                  Cold Side Mass Flow
459.33                  Cold Side Enthalpy
0                       Cold Side State

0.0                     EG Mass flow in Air
0.08                    Tube Diameter
1                       Material
```

Sample Output File (hx.out)

```
Heat Exchanger Output File

-----------------------------------------------------
Property                      Hot Side    Cold Side
-----------------------------------------------------
Temperature (R)               243.282     251.565
Pressure (psia)               11.3714     796
Mass Flow (lb/min)            5924.34     1500
Enthalpy (R)                  -84.1652    694.435
Specific Heat (Btu/lb/R)      0.240765    3.32604
Density (lb/ft3)              0.0655553   0.822179
State (0=gas,1=liquid)        0           0


Effectiveness                 0.8         0.0331857
```

```
Volume (ft3)                  124.599
Area (ft2)                    15.0619
Length (ft)                   8.27245
Number of tubes               470
Cost Units                    81888.6
Reliability Index             0.00328
Core weight (lbs)             21530.7
HEX Weight (lbs)              39369.5
Error flag                    0
```

HX.fileWrapper

```
# @author: John Crowley
# @version: 3
# @description: ALS HEX wrapper

RunCommands
{
    generate HXin
    run "HX.exe"
    parse HXout
}


RowFieldInputFile HXin
{
    templateFile: hx.inp.template
    fileToGenerate: hx.inp

    setGroup HotSideIN
    variable: T       double  6   1   description="Incoming air temperature"
                      lowerBound=0.00 units="deg R"
    variable: P       double  7   1   description="Incoming air pressure" units="psia"
                      lowerBound=0.00
    variable: Mdot    double  8   1   description="Incoming air Mass flow"
                      lowerBound=0.00 units="lb/min"
    variable: H       double  9   1   description="Incoming air Enthalpy"
                      units="Btu/lb/R"
    variable: State   integer 10  1   description="Incoming air state" enumValues="0,1"
                      enumAliases="Gas,Liquid"
    variable: e       double  11  1   description="Air side effectiveness"
                      lowerBound=0.00 upperBound=1.00
    variable: v       double  12  1   description="Incoming air velocity" units="ft/s"
                      lowerBound=0.0
    variable: EGmdot  double  20  1   description="Mass flow of ethylene glycol"
                      units="lb/min" lowerBound=0
    setGroup ColdSideIN
    variable: T       double  14  1   description="Incoming LH2 temperature"
                      lowerBound=0.00 units="deg R"
    variable: P       double  15  1   description="Incoming LH2 pressure" units="psia"
                      lowerBound=0.00
    variable: Mdot    double  16  1   description="Incoming LH2 Mass flow"
                      lowerBound=0.00 units="lb/min"
    variable: H       double  17  1   description="Incoming LH2 Enthalpy"
                      units="Btu/lb/R"
    variable: State   integer 18  1   description="Incoming LH2 State" enumValues="0,1"
                      enumAliases="Gas,Liquid"
    setGroup SizingIN
    variable: tube_diameter   double  21  1   description="HEX Tube Diameter" units="ft"
                      lowerBound=0.00
    variable: Material int    22  1   description="Material" enumValues="1,2"
                      enumAliases="Aluminum,Stainless Steel"
}


RowFieldOutputFile HXout
{
    fileToParse: hx.out
    setDelimiters
    setGroup HotSideOUT
    variable: T       double  6   3   description="Outgoing air temperature" units="deg R"
    variable: P       double  7   3   description="Outgoing air pressure" units="psia"
    variable: mdot    double  8   4   description="Outgoing air mass flow" units="lb/min"
    variable: H       double  9   3   description="Outgoing air enthalpy"
                      units="Btu/lb/R"
    variable: Cp      double  10  4   description="Air specific heat" units="Btu/lb"
    variable: rho     double  11  3   description="Air density" units="lb/ft3"
    variable: State   integer 12  3   description="Outgoing air state" enumValues="0,1"
                      enumAliases="Gas,Liquid"
    variable: e       double  15  2   description="Actual hot Side effectiveness"
    setGroup ColdSideOUT
    variable: T       double  6   4   description="Outgoing LH2 temperature" units="deg R"
    variable: P       double  7   4   description="Outgoing LH2 pressure" units="psia"
    variable: mdot    double  8   5   description="Outgoing LH2 mass flow" units="lb/min"
    variable: H       double  9   4   description="Outgoing LH2 enthalpy"
                      units="Btu/lb/R"
    variable: Cp      double  10  5   description="LH2 specific heat" units="Btu/lb"
    variable: rho     double  11  4   description="LH2 density" units="lb/ft3"
    variable: e       double  15  3   description="Cold side effectiveness"
    variable: State   integer 12  4   description="Outgoing LH2 state" enumValues="0,1"
                      enumAliases="Gas,Liquid"
    setGroup SizingOUT
    variable: V       double  16  3   description="HEX Volume" units="ft3"
    variable: A       double  17  3   description="Cross-sectional area" units="ft2"
    variable: Length  double  18  3   description="HEX length" units="ft"
    variable: Tubes   int     19  4   description="Number of tubes"
    variable: CU      double  20  3   description="Cost units" units="?"
    variable: RI      double  21  3   description="Reliability Index" units="?"
    variable: wt_core double  22  4   description="Core weight" units="lbs"
    variable: wt_hex  double  23  4   description="Overall HEX weight" units="lbs"
    variable: error_flag      int     24  3   description="Error flag" enumValues="0,1"
                              enumAliases="No error,Insufficient Mass Flow"

}
```

Water Separator Component in C++

As with the heat exchangers, the simplest way to use the water separator component is to run its executable. The component WS.exe was written in C++ and relies on simple file based input and output, and can be executed on any Windows operating system. Source code is available for compiling for different operating systems; it has been successfully compiled with Borland's BCC55 for Windows execution and the Unix compiler g++.

Shown below is the notional input file for the ALSWS component. The first five lines are merely a header; these headers can be changed as desired as long as the number of lines before the actual data begins remains constant. Both the executable and the ModelCenter fileWrapper depend on knowing where the data are. The input file is fairly self-explanatory, but the entries in it will be delineated here.

Notional Input File

```
Header1

----------------------------------------
Header2
----------------------------------------
TI
T
```

```
PI
P
MDOT
EGMDOT
H
RHI
ETA
STATE
```

Table A3 lists all of the variables along with their properties. The three variables ending with "I" are reference variables; TI, PI, and RHI describe air properties at which the relative humidity of the air stream is known. This applies to the air entering the inlet in most cases. STATE is an integer variable that describes the state of the air entering the separator. It is mostly used as a flow variable to be passed to heat exchanger components, but will return an error if the air is a liquid (STATE=1). A sample input file can be found at the end of this section.

Table A3: ALSWS System Inputs

Variable Name	Description	Units	Type	Lower Bound	Upper Bound
TI	Reference Temperature of Air	deg R	Double	0	--
T	Incoming Air Temperature	deg R	Double	0	--
PI	Reference Pressure of Air	psia	Double	0	--
P	Incoming Air Pressure	psia	Double	0	--
MDOT	Mass Flow of Incoming Air	lbm/min	Double	0	--
EGMDOT	Mass Flow of Ethylene Glycol Added to Air	lbm/min	Double	0	--
H	Incoming Air Enthalpy	Btu/lbm	Double	--	--
RHI	Reference Relative Humidity of Air	--	Double	0	1
ETA	Water Separator Efficiency	--	Double	0	1
STATE	Incoming Air State	Liquid, Gas	Integer	0	1

The executable is run using the command WS.exe (or WS if it has been compiled using the g++ command line g++ ws.cpp –o WS). This will produce the output file. If using the ModelCenter version of the WS component, the program will be executed automatically by running the component. The output file has the notional format below:

Notional Output File

```
Water Separator Output File

--------------------------------------------------
Property                                    Value
--------------------------------------------------
TOUT
```

TDEW
POUT
MOUT
HOUT
PVDEW
PVDRY
HRIN
HROUT
RHOUT
DELTAHR
HRSAT
WDRAIN
CU
DR
RI
V
WT
WTERROR
WTI
STATE
ERRFLAG

Again, the properties of these variables are listed below in Table A4. A number of the output variables are the outflow of input variables. However, it also outputs several variables describing the humidity of the air stream before and after separation. PVDEW and PVDRY are the vapor pressures at the dew point and the incoming air temperature. HRIN and HROUT are absolute humidities in lbm water/lbm air. If they are the same, no water is removed and DELTAHR and WDRAIN will be zero. HRSAT is the saturation humidity of the air stream at its specified temperature. The variables CU and RI represent cost units and reliability index respectively, but the relationships providing these values did not provide units; therefore, they should only be used to gauge relative prices of different water separator designs. ERRFLAG indicates if there was an error in the component; a value of 0 indicates no errors, a value of 1 indicates that the air was already liquid when it entered the separator, and a value of 2 indicates that the water has frozen. A sample output file is shown at the end of the section.

Table A4: ALSWS System Outputs

Variable Name	Group	Description	Units	Type
TOUT	Performance	Outgoing Air Temperature	deg R	Double
TDEW	Performance	Dew Point of Incoming Air	deg R	Double
POUT	Performance	Outgoing Air Pressure	psia	Double
MOUT	Performance	Outgoing Mass Flow	lbm/min	Double
HOUT	Performance	Outgoing Enthalpy	Btu/lbm	Double
PVDEW	Performance	Vapor Pressure at Dew Point	psia	Double
PVDRY	Performance	Vapor Pressure at Incoming Air Temperature	psia	Double
HRIN	Performance	Absolute Humidity of Incoming Air Stream	lbm water/lbm air	Double
HROUT	Performance	Absolute Humidity of Outgoing Air Stream	lbm water/lbm air	Double
RHOUT	Performance	Outgoing Relative Humidity	--	Double
DELTAHR	Performance	Change in Absolute Humidity	lbm water/lbm air	Double
HRSAT	Performance	Absolute Humidity of Saturated Air	lbm water/lbm air	Double
WDRAIN	Performance	Water Removed From Air Stream	lbm/min	Double
CU	Sizing	Cost Units	?	Double
DR	Sizing	Development Risk	?	Double
RI	Sizing	Reliability Index	?	Double
V	Sizing	Volume	ft3	Double
WT	Sizing	Weight	lbm	Double
WTERROR	Sizing	Weight Error	lbm	Double
WTI	Sizing	Installed Weight Factor	lbm	Double
STATE	Sizing	State	Liquid,Gas	Integer
ERRFLAG	Sizing	Error flag	No error,Air Liquefied,Water Frozen	Integer

Water Separator Operation in ModelCenter

To operate the ALSWS component in ModelCenter, a fileWrapper was created. Filewrappers are stored on an Analysis Server and tell ModelCenter where to find input and output variables. The fileWrapper, seen at the end of this section, details the input, execution, and output of the heat exchanger component. The variables are indexed by row and field number in the input and output files. For more information on the operation of fileWrappers, please consult the ModelCenter documentation.

To load the ALSWS component, simply connect to an Analysis Server that hosts the ALSWS fileWrapper, executables, and input/output files. Drag the component into the project window, and the component will automatically load. Input variables are displayed by icons with green arrows heading into a box, while output variables are displayed by icons with blue arrows heading out of the box. The output variables are grouped by the fileWrapper to help organize them. Output groups include the following variables:

Performance:	Contains all outputs applying to air properties such as temperature, pressure, and humidity properties
Sizing:	All weights, dimensions, and the error flag are found in this group

To execute the model, either right click on the component and select Run, or click on the arrow in the upper left hand corner of the component in the Project Window.

Water Separator Operation in Model Engineer

To operate the ALSWS component in Model Engineer, Visual Basic, the Model Engineer libraries, and the ALSWS component must be installed and registered on your system. Open Visual Basic, and start a new project (Standard EXE). A form will be created. Hit Alt-P or click on "Project" in the Visual Basic Toolbar and hit Ctrl-T or select "Components…" to open the Components window. Scroll through the available components until you find ALSWS. See the Model Engineer Heat Exchanger section for figures displaying these steps. By clicking and dragging this icon to the form, the ALSWS component can be added to the form.

As before, there are two ways to edit the inputs and view the outputs of the ALSWS component. First, you may use the Properties window and select the ALSWS component from the drop down menu. The variables of the component will be listed below, and may be edited by clicking in the appropriate box and typing in a new number. Secondly, you can right click on the component itself and select "Edit". Then right click again and select "Properties" to open the component. There are three property pages; the first simply contains information for Model Engineer. The next two pages contain performance and sizing properties respectively. Any variable boxes colored blue are read only, but any other variable boxes may be edited. To see the results of an input change, simply press the Apply button and the output values will update automatically.

Sample Water Separator Input/Output/FileWrapper Files:

Sample Input File (ws.inp)

```
Water Separator Input File

-----------------------------------------
Value
-----------------------------------------
498.21          Ti
449.919         T
13.2668         Pi
13.2668         P
6000.0          Air stream mass flow
12.0            EG mass flow
-21.7276        H
1.0             RHi
0.85            Efficiency
0               State (0=gas, 1=liquid)
```

Sample Output File (ws.out)

```
Water Separator Output File

-----------------------------------------------------
Property                                  Value
-----------------------------------------------------
Temperature (R)                           449.919
Dew Point Temperature (R)                 449.914
Pressure (psia)                           12.6035
Mass Flow (lb/min)                        5974.83
Enthalpy (Btu/lb)                         -21.7276
Vapor Pressure Tdew (psia)                0.01375
Vapor Pressure Tdry (psia)                0.01375
Humidity IN (lb H2O/lb air)               0.00543516
Humidity OUT (lb H2O/lb air)              0.00121765
Relative Humidity OUT                     1
delta Humidity (lb H2O/lb air)            0.0042175
Saturated Humidity (lb H2O/lb air)        0.000473388
Water Removed (lb/min)                    25.1682
Cost Units (?)                            1186.95
Development Risk (?)                      1
Reliability Index                         0.00285
Volume (ft3)                              980639
Water Separator Weight (lbs)              561.6
Weight error (lbs)                        97.1568
Installed Weight Factor                   115.128
State                                     0
Error Flag                                0
```

WS.fileWrapper

```
# @author: John Crowley
# @version: 3
# @description: ALS Water Separator wrapper

RunCommands
{
    generate WSin
    run "WS3.exe"
    parse WSout
}


RowFieldInputFile WSin
{
    templateFile: ws2.inp.template
    fileToGenerate: ws.inp

    variable: Ti      double  6   1   description="Reference air temperature"
                      lowerBound=0.00 units="deg R"
    variable: T       double  7   1   description="Incoming Air temperature"
                      lowerBound=0.00 units="deg R"
    variable: Pi      double  8   1   description="Reference air pressure" units="psia"
                      lowerBound=0.00
    variable: P       double  9   1   description="Incoming air pressure" units="psia"
                      lowerBound=0.00
    variable: Mdot    double  10  1   description="Incoming air Mass flow"
                      lowerBound=0.00 units="lb/min"
    variable: EGmdot  double  11  1   description="Ethyelene glycol mass flow"
                      lowerBound=0.00 units="lb/min"
    variable: H       double  12  1   description="Incoming air Enthalpy"
                      units="Btu/lb/R"
    variable: RHi      double  13  1   description="Reference Relative Humidity"
                       lowerBound=0.00 upperBound=1.00
    variable: e        double  14  1   description="Separator Efficiency" lowerBound=0.00
                       upperBound=1.00
    variable: State    integer 15  1   description="State of fluid" lowerBound=0
                       upperBound=1 enumValues="0,1" enumAliases="Gas,Liquid"
}


RowFieldOutputFile WSout
{
    fileToParse: ws.out
    setGroup Performance
    variable: T        double  6   3  description="Outgoing air temperature" units="deg R"
    variable: Tdew     double  7   5  description="Dew Point Temperature" units="deg R"
    variable: P        double  8   3  description="Outgoing air pressure" units="psia"
    variable: PvTdew   double  11  5   description="Vapor Pressure at Tdew" units="psia"
    variable: PvTdry   double  12  5   description="Vapor Pressure at Tdry" units="psia"
    variable: mdot     double  9   4   description="Outgoing air mass flow" units="lb/min"
    variable: Wdrain   double  18  4   description="Water removed" units="lb/min"
    variable: H        double  10  3   description="Outgoing air enthalpy"
                       units="Btu/lb/R"
    variable: HRin     double  13  6   description="Absolute incoming Humidity" units="lb
                       water/lb air"
    variable: HRout    double  14  6   description="Absolute outgoing humidity" units="lb
                       water/lb air"
    variable: RH       double  15  4   description="Relative outgoing humidity"
    variable: deltaHR  double  16  6   description="Change in absolute humidity" units="lb
                       water/lb air"
    variable: HRsat        double  17  6   description="Saturated Humidity" units="lb
                           water/lb air"

    setGroup Sizing
    variable: V        double  22  3   description="Separator Volume" units="ft3"
    variable: CU       double  19  4   description="Cost units" units="?"
    variable: DR       double  20  4   description="Development Risk" units="?"
    variable: RI       double  21  3   description="Reliability Index" units="?"
    variable: wt_ws    double  23  5   description="Overall Separator weight" units="lbs"
    variable: wt_d     double  24  4   description="Separator weight error" units="lbs"
    variable: wt_i     double  25  4   description="Installed Weight Factor"
    variable: State    integer 26  2   description="State of outgoing fluid"
                       enumValues="0,1" enumAliases="Gas,Liquid"
    variable: errorFlag       integer 27  3   description="Error flag" enumValues="0,1,2"
                              enumAliases="No errors,air liquefied,waterfrozen"
}
```

Creating System Models in ModelCenter

Creating models comprised of the heat exchanger and water separator components is a simple matter of loading each individual component and then linking them together. For each component in a system, a separate component should be added to the Project Window. The Link Editor should then be opened in ModelCenter and links created. There are two approaches that may be used to create system models: Brute-Force, and Optimizer Based Decomposition (OBD). OBD is desirable from a Multidisciplinary Design Optimization standpoint, but takes longer to set up and it is suggested that the user has a good understanding of the system and components involved when trying to create an OBD model.

The Brute-Force method involves linking every flow variable possible, which will in effect cause feedback loops to occur. As the properties of the fluids running through the heat exchangers change, the components will have to be reexecuted to account for changes to the air and hydrogen properties. But once the components are executed again, the properties will have changed. This process iterates until there is no longer a significant change between executions and the model stops at the solution.

To create a system model that employs feedback, all temperature, pressure, enthalpy, state, velocities, and mass flow variables should be linked. However, the mass flow inputs for the cold (hydrogen) side may be left unlinked if you wish to remove some hydrogen mass flow as it moves through the system. It is generally not advisable to do this, as the mass of hydrogen entering a component should always be equal to or less than the preceding component; removing hydrogen and then adding it back again later will not be modeled properly as the two hydrogen streams will have different properties. Links are created in ModelCenter's Link Editor; simply drag an output variable from the left on top of the appropriate input variable on the right. Once the links are created, the option to manually enter the values for linked input variables will no longer exist.

With the links fully defined, it simply remains to enter all the input values. This includes tube sizes, heat exchanger effectiveness values, material selections, ethylene glycol mass flow, separator efficiencies, and reference humidity values. If reference temperatures and pressures in the water separator components are not linked, these must also be specified. Once all inputs have been specified, click on the Run Scheduler to automatically iterate the model until it has converged.

The OBD method will require fewer runs of the components on the whole, although the quick execution of the ALS components mitigates this issue somewhat. To create an OBD scheme, an optimizer will have to be employed to maintain compatibility constraints between the cold side heat exchangers. There are no longer any feedbacks, so the only links are those between the hot side inputs and outputs. Links should be created between the temperature, pressure, mass flow, enthalpy, and state variables of all hot side inputs.

Now constraints must be defined. Blank script components are found in ModelCenter's Common/Functions directory. Between every heat exchanger, there must be compatibility constraints for temperature and pressure. For a number N heat exchangers, there will 2*(N-1) compatibility constraints. To calculate these constraints, the output and input cold side temperatures and pressures should be input into the script. For a simple two heat exchanger system consisting of a precooler and condenser, for example, there would be two compatibility constraints calculated by:

Tconstraint = Precooler.ColdSideIN.Temperature-Condenser.ColdSideOUT.Temperature
Pconstraint = Precooler.ColdSideIN.Pressure-Condenser.ColdSideOUT.Pressure

It is advisable that these constraints be normalized and then squared; in this manner, the compatibility constraint can be reflected as inequality constraints rather than equality constraints. When fed to an optimizer, the only way for this constraint to be satisfied is for T_{in} to be equal to T_{out}. By normalizing the magnitude of one constraint cannot dominate over another.

Other constraints may be necessary depending on the system being analyzed. If there are conditions at the exits of either the hot or cold sides that must be met these should be represented as inequality constraints as well. Additionally, if cold side mass flows are allowed to be different between heat exchangers, the constraints should be added that mass flow can only decrease as the hydrogen moves through the system, never increase.

With the constraints defined, the optimizer that will drive the OBD can be added. ModelCenter includes a default optimizer in its Common/Drivers directory. Any variable or combination of variables can be optimized as long as an appropriate objective function is defined. To optimize total system weight, for example, create a script that adds all the individual component weights together, and use the output as the objective function. The constraints defined in the constraint script should all be dragged in to the constraint window. Finally, the design variables must be chosen.

All of the input variables that defined the compatibility constraints must be used as design variables to allow those constraints to be met. Any other design variables should be added as appropriate; generally, the mass flows of hydrogen and the heat exchanger effectiveness values should be design variables, as they are major drivers of the system.

Creating System Models in Model Engineer

Creating a system model in Model Engineer can be a more difficult proposition if the Fixed Point Iteration approach is to be taken. The Brute-Force method is easily achievable, however. To create a Model Engineer ALS model, a new Visual Basic Standard EXE project should be started as normal. The components must be added to the form as described in the heat exchanger and water separator sections of this document. Additionally, the Modelogics Data Flow and Modelogics Data Viewing libraries should be added.

To connect the components in the model, the Connector Arrow is dragged from component to the other. Model Engineer should automatically identify those variables which are to be connected, but within the connector window the link direction and individual links may be changed. Direct links from one component to another should be made only between hot side variables; this includes links to any water separators. To create the feedback links for the cold side variables, the Splice component from the Data Flow library should be added in between each heat exchanger. The cold side outputs can be routed through this component and to the cold side inputs of the next component.

To execute the model, a DataGenerator component from the Data Viewing library must be added. By right clicking on it and selecting “Properties”, the execution order of the component can be set. The system should execute in the order that the hot air goes through the system. Make sure to check the box next to each component to ensure that it runs.

If desired, a schematic viewer can be added that allows the user to see the system variables changing as the model executes. To create a schematic viewer, the S-Viewer component should be added from the Data Viewing Library. Next, a drawing of the system can be created in Microsoft Power Point, although this is optional. The Power Point drawing should be saved as a Windows Metafile (*.wmf). By right clicking on the S-Viewer component and selecting “Properties”, the *.wmf file can be added to the component and will be displayed when the S-Viewer is clicked during execution. In the S-Viewer property pages, the variables to be displayed on the picture are selected. These variables can be arranged on the drawing (or white space if no *.wmf file is uploaded) by right clicking on the S-Viewer component and selecting “Show Viewer”.

To execute this model, it should either be compiled into an executable, or run directly from Visual Basic by going to the Run menu and clicking Start (or hitting the Start button on the toolbar). The model should iterate continuously until the inputs and outputs between the components all match. This process executes very rapidly, but does involve a lot of extra computations.

To use the FPI method of bringing a closed solution, the hot side Connector Arrows are still used, but the cold side links are removed. Instead, an Excel file is used along with the Model Engineer components ReadXCells and WriteXCells to continuously update the cold side properties based on the results of the previous run. Essentially, each run takes up a column in the Excel file. Initial conditions for all variables are specified in the first column. When the model is executed, it will write the cold side outputs between each heat exchanger in the same column in the outputs section. These values are written in the next column’s input section via logic in the worksheet itself. By employing a Do Loop component, the Data Generator can be told to execute a set number of times, indexed by the current column. Each iteration reads from and writes to the next column for however many iterations are specified. At the end of the process, the closeness of the cold side guesses to the actual values can be verified. If there is sufficiently small error between the two, the system has been closed. Otherwise, the model should be executed again with the most recent inputs in the first column until it has satisfied an appropriate maximum error condition.

Please note that the Model Engineer models cannot truly be optimized, they can only be closed for certain inputs. The optimization capability of Model Engineer is too simple to accommodate the complex optimization strategy necessary to tackle an ALS of any size.