Operation Manual

Team #2

Decreasing Electricity Usage at Cummins Technical Center

Members

Daniel Baker
(dpb11f@my.fsu.edu)
Warren Bell
(wab10@my.fsu.edu)
Daniel Carnrike
(dac10c@my.fsu.edu)
Kyle Fields
(krf11b@my.fsu.edu)
Marvin Fonseca
(mjf12g@my.fsu.edu)

Advisors/Sponsors/Faculty

Dr. Ordonez, Dr. Hays
Cummins Inc.
Dr. Gupta

04/03/15

Table of Contents

Table of Figures

Table of Tables

Abstract

Team 2 created the following operation manual in order to assist a user in adjusting different parameters in order to have a variety of system outputs. Some parameters that the user can alter are: the number of engines running and the date that the system is running and by having these parameters altered, any user can have the best available simulation for Team 2's exhaust gas power generation system. A function analysis for the proposed design and simulation can be found within the report. The function analysis describes the conditions and process Team 2's system undergoes and the function of the guided user interface (GUI) using the MATLAB coding program. The GUI allows for the user to adjust different parameters with ease and without previous coding knowledge. The GUI displays different output parameters that the user may select from a preset list of outputs in a drop down menu. Some output parameter options are: total power output, exit temperature of the solar collectors, insolation and a cost analysis graph. The cost analysis graph output option takes into account any regular maintenance the system will need and is described in the later part of this report. The following report also describes any problems that arise from running the simulation and how to correct such problems. An important factor that should be mentioned about Team 2's exhaust gas power generation system is that the components described are theoretically based, as the team does not have the budget to purchase and build this multi million dollar system.

Acknowledgements

Team 2 would like to personally thank Dr. Hays, Dr. Ordonez, and Dr. Gupta for their guidance and expertise that they provided throughout the duration of this project.

1. Introduction

Energy efficiency is a major concern for many businesses, especially for industrial businesses. In order to reduce energy consumption, the replacement of inefficient light bulbs, poor insulation, and piping are all fundamental ways to reduce energy consumption. Techniques used to reduce electricity usage range from implementing motion sensor lighting, maintaining up to date insulation and upgrading components to state of the art parts. Reducing energy consumption is a small step in making industrial companies more eco-friendly. Cummins, Inc. has made great strides to become an industry leader in recyclable energy practices that minimize impact on the environment. In 2011, Cummins, Inc. began to supplement some of its power sources in certain locations with solar panels and rerouting energy developed from their test cell dynamometers back into the grid. Also, Cummins, Inc. had an energy audit conducted for their technical center in order to reduce their initial energy consumption. Cummins Technical Center (CTC) is a large engine testing facility located at Columbus, Indiana. The facility is capable of testing 96 Cummins engines which contain many different engine families such as the ISB and ISX engine families. Engine applications for these specific engine families range from pick-up trucks to large train engines. Cummins, Inc. is at the forefront in making industrial technical centers more eco-friendly and less harmful to the environment.

Cummins, Inc. chose to sponsor a senior design team from Florida State University to further reduce their energy consumption by 10%. In order to reduce the CTC energy consumption, Team 2 has been assembled to generate the necessary ideas, conduct the required analysis and produce an energy savings plan. Team 2 focused on the development of multiple energy reduction areas for the CTC and created different ideas; these ideas can be referenced to previous reports such as Final Report I and can be located on the official website. Instead of trying to implement additional energy saving ideas for the spring semester, Team 2 focused on one major idea: capturing exhaust gases created by different engine families tested at the technical center. The following report contains an operation manual for the simulation package generated, in order to determine an overall power generated and system efficiency. The report describes the necessary procedure and assumptions Team 2 diligently created in order to produce the following simulation package. The following report contains: a functional analysis, component specifications, product assembly, operation instructions, troubleshooting and regular maintenance required of the proposed design and simulation package.

2. Background and Function Analysis

The following section contains a background section which explains assumptions made and provides a functional analysis section for the simulation package and proposed design. In order to understand the proposed power generation system Team 2 has designed, a brief background explanation is required for the system. To begin, the entire system that is being simulated is for an Organic Rankine Cycle (ORC) with several different components including a compressor, solar thermal collectors that act as preheaters, a shell and tube heat exchanger, and a turbine that uses n-butane as a working fluid that gains heat from both the sun and the exhaust gases emitted from the technical center.

In order to model this in the simulation, the code works through multiple script files that are called into a main SIM.m function file. There are several function files designed to evaluate the different thermodynamic equations needed to design a thermal fluid system. These coding files are broken into subsections for each component of the system. In particular, the solar thermal collectors needed an extensive amount of coding to account for the variation of several different parameters that affect the energy that can be collected from the sun throughout the day from sunrise to sunset. The results of the day are sent as outputs to the main SIM.m file that incorporates the outputs into the heat analysis in the rest of the components. Just as well, there is a file that contains the results of everyday of the year stored into large matrices that can display the data for each day of the year.

There are several assumptions that needed to be made in order to make sure that the system works properly, such as selecting output pressures and temperatures in order to iteratively solve for the true pressures and temperatures. The design parameters were taken to be based on the sponsor recommendations of 50 test cell engines running 24 hours, 365 days a year in order to design the heat exchanger for the system. Additionally, the properties of the exhaust gas were approximated to the exhaust properties of air, an exhaust temperature of 400^{O}F, and the values can be easily changed to reflect whatever the true properties may be by editing the values found in the SIM.m file if required by the user. The team used reasonable estimates in order to calculate the pressure drops throughout the system using a separate pressure drop file. The SIM.m function file works in tandem with the Guided User Interface (GUI) designed by the team to aid the user in viewing the results of the simulation. In order for the GUI to work properly, it is necessary for all of the files included in the simulation package to be in the same working directory in order to make the program call the correct functions in the script files.

The GUI is intended to make selection of different outputs very simple for the user by using a drop down list of available plots that the user can compare against different numbers of engines being tested and the day of the year. In addition to this, the user can toggle the plots to compare multiple simulation results on the same figure in order to see how changing the different parameters will affect the results. In the GUI, the user simply clicks on the date push button to bring up a calendar menu to select a day of the year for analyzing how the different parameters vary from the time of sunrise to sunset throughout the day. Additionally, the user can manually enter the amount of engines running for several engine families tested by Cummins, Inc. at the facility. This allows a user even with very little coding experience the ability to find meaningful results such as power output and cost analysis of installing an Organic Rankine Cycle at the Cummins Technical Center. The full functionality of the GUI may be found in the Operation Instruction section of this report.

3. Product Specifications

The product specifications described are for Team 2's exhaust gas power generation system. System specifications are based off of individual components that create the overall system. These components consist of: a shell and tube heat exchanger designed by the team, HG601BX Corken Compressor, Siemens SST-060 Series Steam Turbine and AE 3000 Solar Thermal Collectors. The system operates on two separate cycles, an exhaust cycle and an ORC. The exhaust cycle will have an exhaust mass flow rate depending on the number of engines running as explained in the previous section. N-butane was chosen as the working fluid for the ORC system and the mass flow rate was optimized and set to $7\,\frac{kg}{s}$ in order to design an optimum heat exchanger.

The heat exchanger was custom designed by Team 2 in order to generate the required heat transfer from the exhaust system to the n-butane. The heat exchanger is a shell and tube heat exchanger with counter flow. Figure 1 is a detailed drawing of the proposed heat exchanger design.

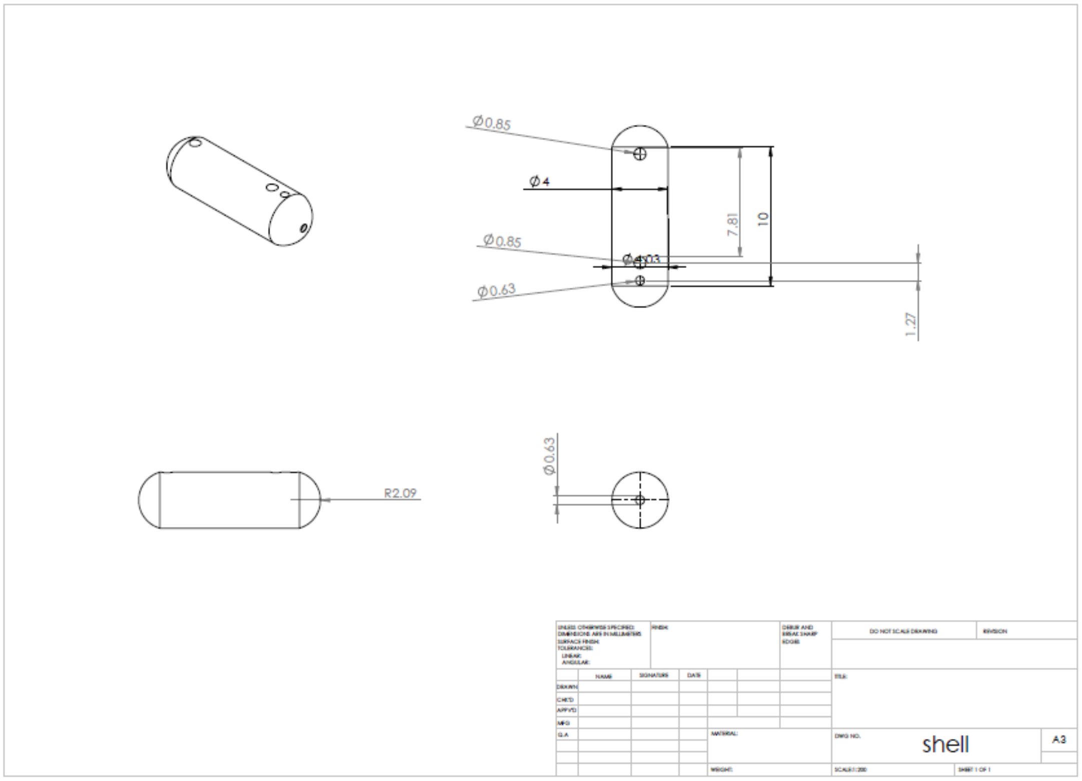


Figure 1: Heat Exchanger Drawing

From Figure 1, the exchanger dimensions are shown and Table 1 is a more detailed representation of the interior of the heat exchanger. The overall length of the heat exchanger is 10m and has a height of 5m.

Table 1: Heat Exchanger Shell Parameters

Insider Shell Diameter	4m
Outside Shell Diameter	4.525m
Number of Baffles	3
Baffle Spacing	3m

The material of the heat exchanger is stainless steel and the tube material is constructed of polished brass. The tube parameters can be seen in Table 2 and the drawing can be located in Appendix A.

Table 2: Tube Parameters

Inner Tube Diameter	245mm
Out Tube Diameter	250mm
Clearance Between Tubes	25mm
Number of Passes	4
Number of Tubes	32

The overall weight of the system for the heat exchanger including the weight of the stainless steel and weight of the tubes equates to 45 Mg. The heat exchanger is capable of transferring 40% of the exhaust energy when Cummins, Inc. has 50 engines running.

The HG601BX Corken Compressor was selected based on its ability to compress n-butane. Also, the compressor is capable of compressing n-butane to 1.1MPa, the required pressure, which is capable of overcoming the pressure losses throughout the system. Figure 2 is a technical drawing for the HG601BX Corken Compressor [1].

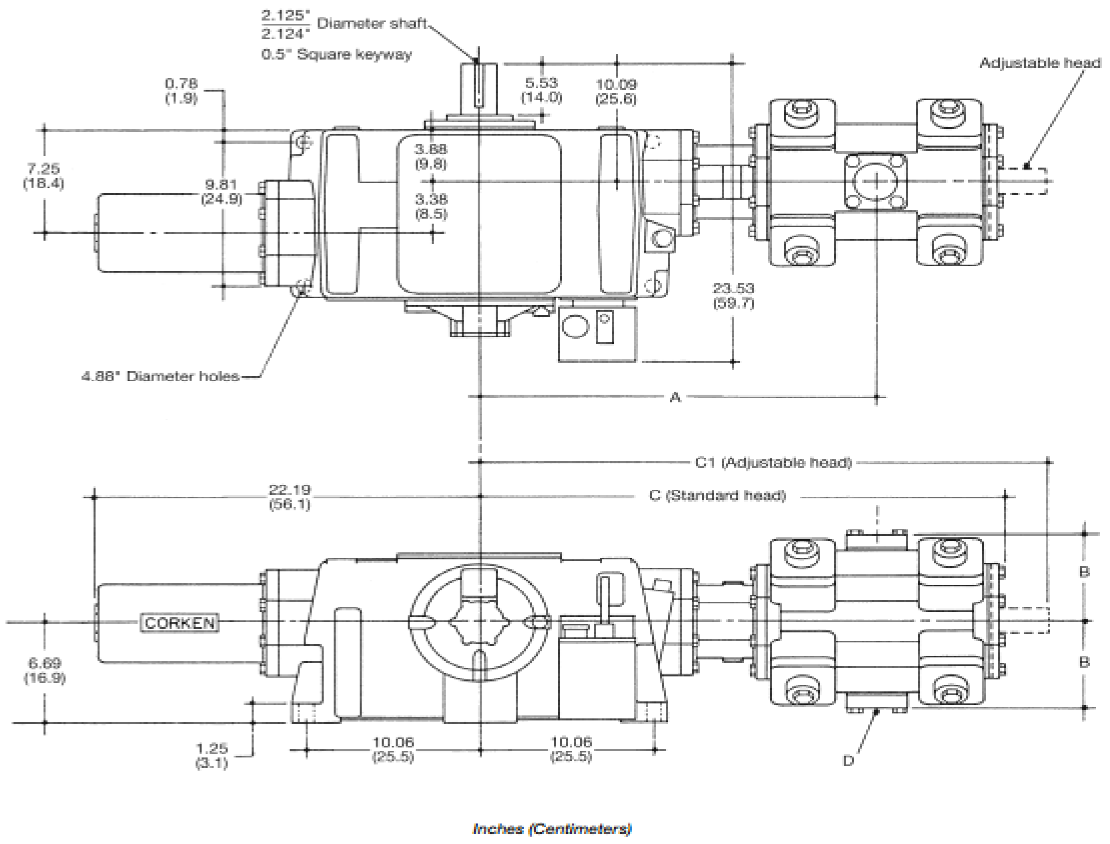


Figure 2: HG601BX Compressor

The weight of HG601BX is 318 kg, operates at 1200 rpm 55 kW, capable of compressing 115cfm of gas and has a 152.4 mm bore. The turbine selected that would extract the maximum amount of work from the incoming n-butane is the Siemens SST-060 Series Steam Turbine. The SST-060 is capable of generating 6MW of electricity. The capable inlet pressure of the turbine is 13.1 MPa. The technical drawings could not be provided by Siemens but the dimensions of the turbine are 1.5m x 2.5m x 2.5m.

AE 3000 Solar Thermal Collectors (STC) was chosen to generate additional heat in the n-butane. The dimensions of one solar thermal collector are 3m in length, 1.2 m in width and 0.076 m in height. Given the total available area at the CTC, 500 collectors would be used for the system. The AE 3000 STCs have a life time of 30 years and are capable of containing 1.22 gals per collector. Depending on the day in the year, the solar collector efficiency range from 26% to 37%. With the varying efficiency of the solar collectors, a temperature output for n-butane range between 90^{O}C to 130^{O}C.

4. Operation Instructions

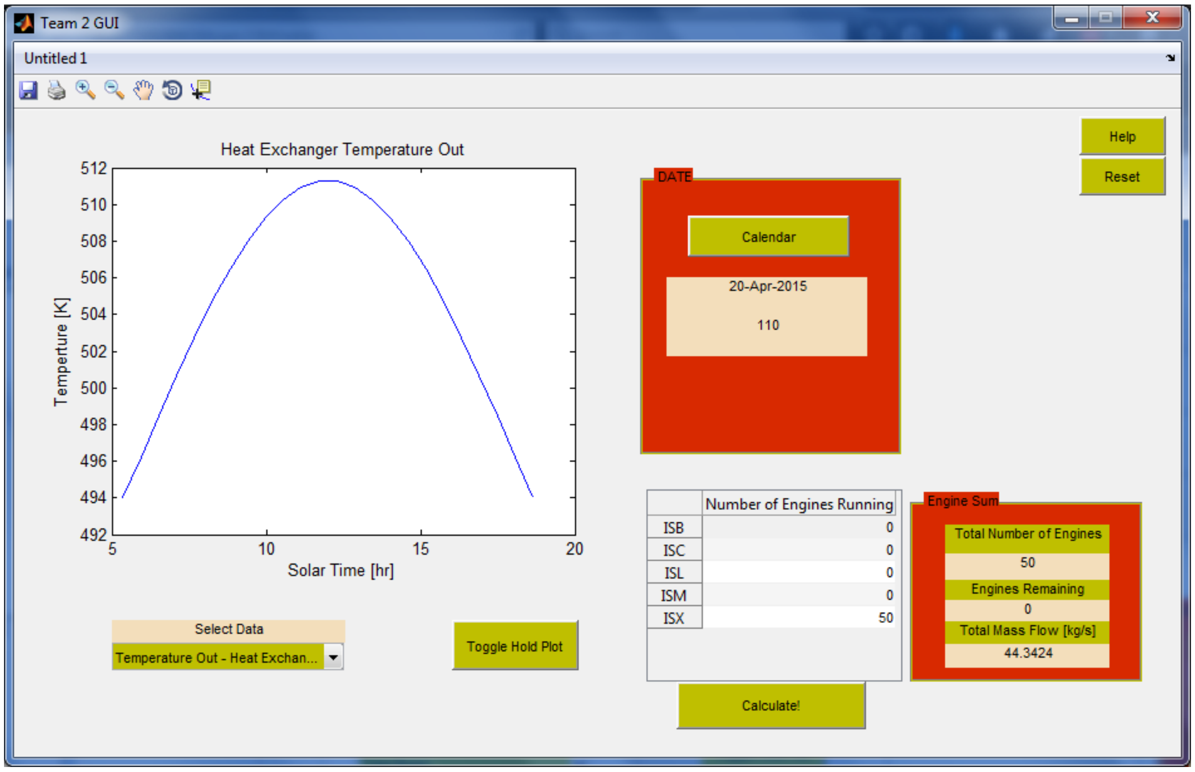
The simulation program created by Team 2 is designed to be a user-friendly program that allows Cummins, Inc. to control different important parameters. The input parameters are the amount of engines running and what day of the year is the system running. The user can then select outputs such as the daily power output, annual power output, and the instantaneous and annual power savings for the system. In order to start the GUI, the user must load the MATLAB program provided by MathWorks. As with any MATLAB program all the files provided by the team must be located within a single working directory. The team has created a zip folder provided for this project that can be saved on the hard drive of the computer that contains all the necessary files. The following files are displayed in Table 3.

Table 3: Required Program Files

Coordinate_Time.m	Cost_Analysis.m	GUI_SIM.fig, GUI_SIM.m
Intersections.m	PipeLoss.m	SIM.asv
SIM.m	Siminstantaneous.m	Solar_Power.xlsx
Solar_Power_2.m	Solar_Thermal_Collectors.m	System_dimension.m
uisedate.m	year.m	

For simplicity and ease to the user, all required files are saved on a zip folder FSU_Senior_Design_Team_2_Cummins_Energy_Savings.zip. Also, these files can be located in Appendix B of this report or downloaded from Team 2's website.

Team 2 built a guided user interface (GUI) using MATLAB. The GUI interface is displayed in Figure 3.



20-Apr-2015
110

	Number of Engines Running
ISB	0
ISC	0
ISL	0
ISM	0
ISX	50

Total Number of Engines
50
Engines Remaining
0
Total Mass Flow [kg/s]
44.3424

Figure 3: Guided User Interface Screen

The main parameters that the user can change are the date and the number of engines running. The user is then able to select various outputs from a dropdown list that shows the plot of the data across a time interval from sunrise to sunset for the selected date.

The program allows the user to specify the amount of engines running for 5 separate engine families: ISB, ISC, ISL, ISM, and ISX. The user clicks on the cell within the table for the entry they would like to edit, and enters a number for the engines running. The program works on the assumption that there will be a maximum of 50 engines running at a given time at the facility, so the sum of the engines running should not exceed this amount. However, the team will give a functionality of the engine cap size to exceed this up to 96 test cells in order to accommodate all test cells located at the CTC.

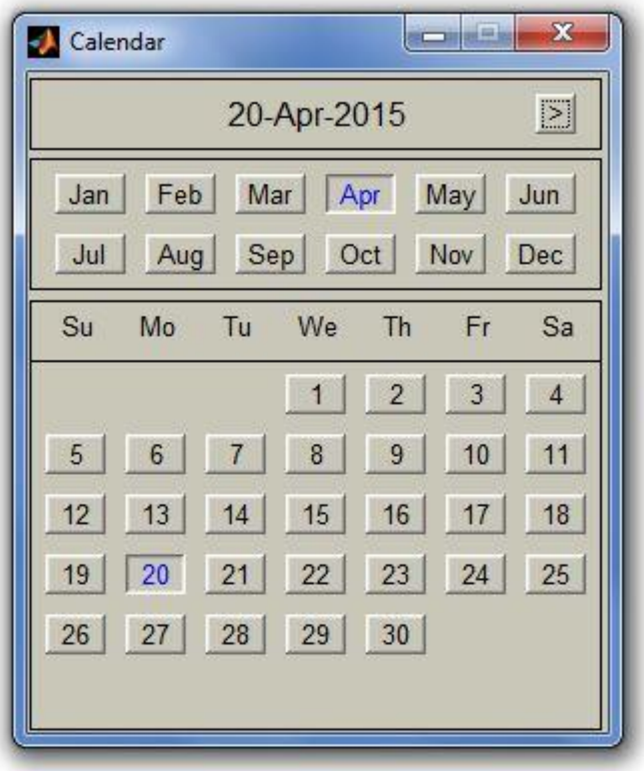


Figure 4: Calendar Input Popup

The second input that the user is allowed to vary is the date. In order to change the date, the user must click on the “Calendar” push button that brings up a calendar popup, shown in Figure 4, wherein the user can select any date for the year. The month and day can be selected by clicking on the toggle button for the desired date. In order to edit the year, the user can strike the “+” or “-“ key to either increase or decrease the year, respectively. It should be noted that the calendar will display the current date of the user’s computer upon the first time it is brought up. Once the date is selected, the user is brought back to the GUI, Figure 3, and the selected date is displayed below the “Calendar” push button. The user can repeatedly select different dates, and the date will be overwritten with each successive selection.

Finally, the plots seen in the graph can be selected from the drop down list located beneath the plot. The options available to user are: the daily power output, annual power output, system efficiency, instantaneous and annual power savings for the system. Along with these, the temperatures and other parameters of the system that vary throughout the day may be selected. Additionally, the cost analysis of the system can be selected. Based on the parameter selected, the plot above the drop down list will change to display the output against the time interval from sunrise to sunset that is determined based on the date selected. The user then has the ability to save, print, zoom in, zoom out, pan, or place a data cursor.

5. Troubleshooting

Once the system is running, the user may run into different problem which this section aims to correct. The primary problem any user will face is the required files labeled in Table 3 are not located within a single working directory. This can be fixed by setting FSU_Senior_Design_Team_2_Cummins_Energy_Savings.zip as the working directory from the MATLAB home screen. The required folder can be downloaded from Team 2's website. Once the working directory is set, the program is operational. Just as well, the spelling of these files is case sensitive, and the files should not be renamed or moved to a separate working directory.

Other problems that could arise from running the program are being trapped in an infinite loop. If the user were to change anything within the code to fit their specific needs and an infinite loop was encountered, pressing ctrl+C while on the MATLAB command window will terminate the current operation, and allow the user to continue editing code without exiting the entire MATLAB program.

A reset button will clear all variables and clear the command window which is located on the top right corner of the GUI simulation. The location of the reset button is clearly displayed in Figure 3. The MATLAB commands "clear all" and "clc" can be typed directly into the command window to clear the variables and the command window if the user chooses to do so. Another helpful tool that can be located on the GUI interface is a help button, which is also highlighted in Figure 3. The help push button provides instructions on how to run the code and is a vital tool for any first time user.

6. Regular Maintenance

The regular maintenance section is intended for the maintenance of the exhaust system power generation system and each individual component that makes up the system. This is more useful for the user instead of maintaining the simulation program which comes down to maintaining a computer. The most important maintenance is performed on the heat exchanger, the cost of maintenance increases exponentially with weeks between cleanings. It is suggested that the heat exchanger is cleaned every 10-12 weeks to maximize efficiency and minimize costs. In this range the cost of each cleaning is in the $50,000 to $60,000 per cleaning totaling to an annual cost of about $275,000.

The Solar Collectors also require maintenance and cleaning. As earlier stated, we have selected the AE series Collectors specifically the AE 3000 system. These collectors operate best when the glass is clean and unobstructed. If the solar collectors become dirty, wash them with mild soapy water and rinse. Remove any branches or leaves that do not naturally fall off or are blown away by the wind. This debris should be minimal due to the fact that the solar collectors are located on the roof. Cost of this cleaning is minimal, but must be monitored throughout the year. These duties can be added to any worker already on staff, which does not require hiring new staff members.

Team 2 has selected the Corken 6 inch single stage horizontal reciprocating Compressor. This compressor uses 7 quarts (6.6 Liters) of oil as lubricant, which must be monitored and changed every 2200 hours of operation along with filter 4225. Piston rings and piston ring expander also must be replaced every 4400 hours of operation (2200 hours if non-lubricated). Packing cups, spacers, O-rings, springs bushing, bearings, and retainer rings would also need multiple replacements within lifespan of the machinery [2]. A more detailed chart is located in Appendix A. Another maintenance requirement for the compressor and turbines is the monitoring of their blades. The compressor and turbine system should be inspected every 8 weeks for any fatigue failures, corrosion effects and regular maintenance. Even though n-butane is a non-corrosive material, routine maintenance checks are required. Many other components can become corroded or cracked and must be monitored. A bi-monthly inspection will consist of a full system cleaning and component analysis. This service requires a team of field engineers to ensure a proper inspection was conducted. The compressor and the turbine can be inspected at the same time to minimize the amount of down time experienced the estimated maintenance cost for the turbine system equates to be $35,555.28 [3].

7. Conclusion

Any user with minimal experience can use the simulation package by following the proposed steps. Also, the user is presented with the required background information to make any adjustments within the simulation code they deem fit. Team 2 designed an exhaust gas capturing system and a simulation code to best simulate how the exhaust gas capturing system will change depending on the number of engines running and the date the system is running. By allowing the user to change these parameters, a more realistic and up to date system can be simulated. The simulation package proposed is designed to change with any technology upgrades for the different components: turbine, compressor, heat exchangers and the solar thermal collector system. A product specification list has been presented to the user in order for the user to better understand what components are considered in Team 2's simulation package. The function analysis is also described to further assist in navigating the GUI and gives the user a better understanding on where the outputs are coming from.

Any malfunctions that may occur while the user is operating the simulation can be corrected by following the trouble shooting section of the report. The GUI is designed to be user friendly. Supplemented with the background information and design for manufacturing, reliability and economics report, any user, regardless of knowledge, can operate Team 2's GUI with the proper guidance and assumptions clarified.

References

[1] "A Global Leader in Industrial Compression & Pumping Solutions." *Industrial Compressor*. Web. 1 Apr. 2015. <http://www.corken.com/Home>.

[2] "Air Compressor Maintenance." *Guide*. Web. 2 Apr. 2015. <http://www.portlandcompressor.com/compressor/maintenance.aspx>.

[3] Darrow, Ken, Rick Tidball, James Wang, and Anne Hampson. "Catalog of CHP Technologies." U.S. Environmental Protection Agency, 1 Jan. 2015. Web. 3 Apr. 2015. <http://www.epa.gov/chp/documents/catalog_chptech_3.pdf>.

Appendix A – Helpful Diagrams

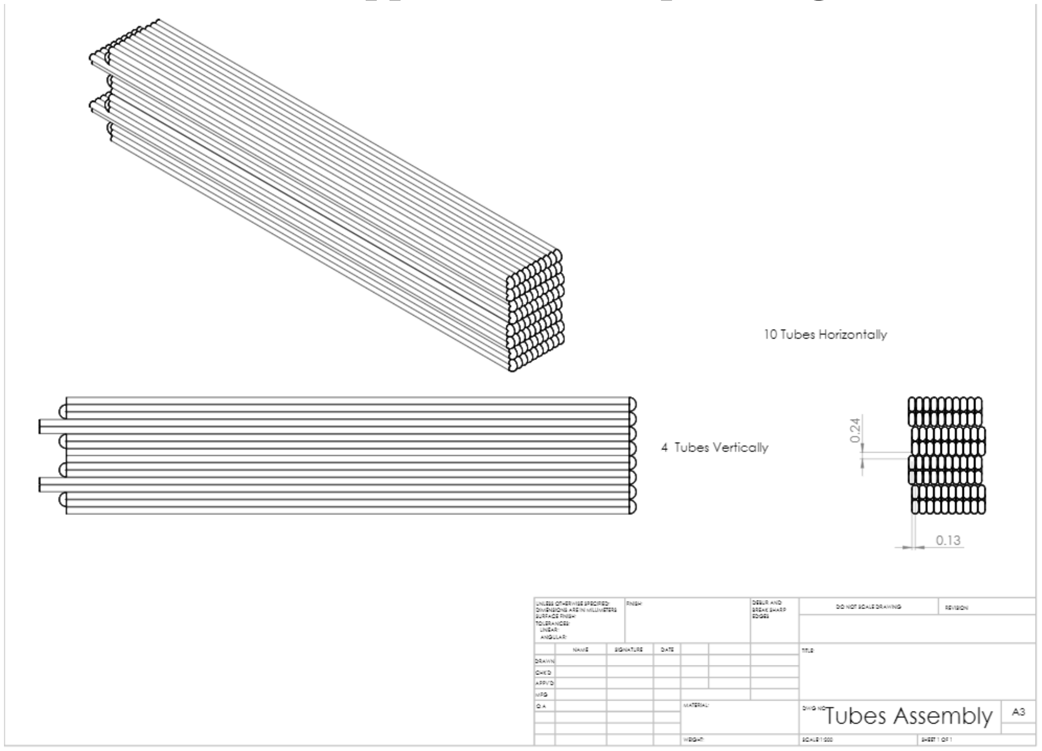
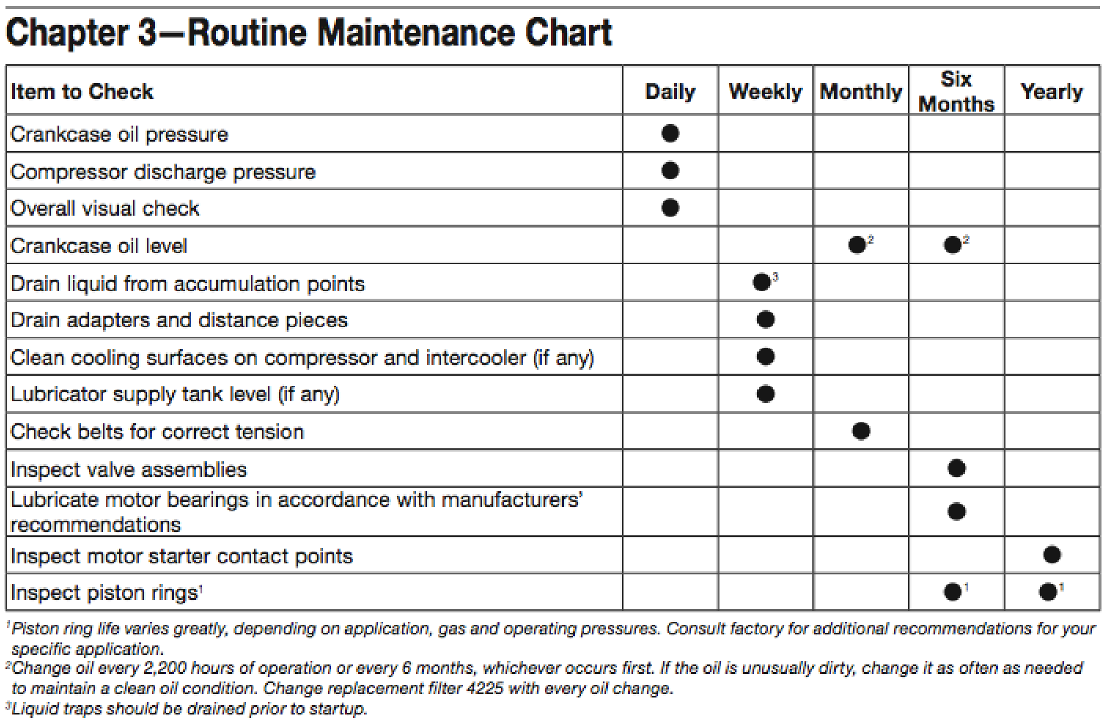


Figure 5: Heat Exchanger Tubes



Chapter 3—Routine Maintenance Chart

Item to Check	Daily	Weekly	Monthly	Six Months	Yearly
Crankcase oil pressure	●				
Compressor discharge pressure	●				
Overall visual check	●				
Crankcase oil level			●[2]	●[2]	
Drain liquid from accumulation points		●[3]			
Drain adapters and distance pieces		●			
Clean cooling surfaces on compressor and intercooler (if any)		●			
Lubricator supply tank level (if any)		●			
Check belts for correct tension			●		
Inspect valve assemblies				●	
Lubricate motor bearings in accordance with manufacturers' recommendations				●	
Inspect motor starter contact points					●
Inspect piston rings[1]				●[1]	●[1]

[1] *Piston ring life varies greatly, depending on application, gas and operating pressures. Consult factory for additional recommendations for your specific application.*
[2] *Change oil every 2,200 hours of operation or every 6 months, whichever occurs first. If the oil is unusually dirty, change it as often as needed to maintain a clean oil condition. Change replacement filter 4225 with every oil change.*
[3] *Liquid traps should be drained prior to startup.*

Figure 6: Compressor Maintenance Chart [2]

Appendix B – MATLAB CODE

```
% Cost Analysis Heat Exchanger
% Senior Design
% Last Updated: 3/22/15

D = 4; % Units [m]
D_old = 5.5;
L = 10; % Units [m]
L_old = 21;

Cs = 1800*(D^2.5)*L; % Cost of the shell % Units [$]
Cf = 350*D*L; % Initial cleaning cost % Units [$]
Ct = 1950; % First cost of the tubing % Units [$]
C = Cs + Cf + Ct

Cs = 1800*(D_old^2.5)*L_old; % Cost of the shell % Units [$]
Cf = 350*D_old*L_old; % Initial cleaning cost % Units [$]
Ct = 1950; % First cost of the tubing % Units [$]
C = Cs + Cf + Ct
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Senior Design
%PipeLosses/Heat losses

clc
clear all

numberofpoints = 20;
e = 0.0046;%Commerical Steel Roughness
PD = 0.5; % [m]
thickness = 0.0307; % [m]
g = 9.81;
nB_density = 2.48; % [kg/m^3] Vapor stage
Leq = 25; % [m]
T_ambient = 20; %Degrees C

%%%Piping Material Properties
k_bearsteel = 43; %W/m*K
cp_nbutane = 2282; %J/kg*K
k_nbutane = 0.0342;


%%%Butane_mass_flow = 40; % [kg/s]
AreaPipe = (PD^2)*(pi/4);
dynamic_viscosity = (0.0000124);%Pa*s
kinematic_viscosity = dynamic_viscosity / nB_density;

%%%%%% Heat Exchanger to Turbine %%%%%%
F = zeros(1,numberofpoints); %rows x columns
P_turbinein = zeros(1,numberofpoints);
%%%VBT = Butane_mass_flow / (AreaPipe * nB_density)
P_boilerout(1:numberofpoints) = 1000000; % PressureofVaporout Pa
Vbt =  linspace(1,25,numberofpoints); %[m/s] Velocity From boiler to inlet of Turbine
%%%%%%%%%%%%%%%%%%%%%
Re = (Vbt.*PD ./ kinematic_viscosity);

i = 1;
j = 1;
n = 1;

%Finding the Frictional Factor for different Re #
while i <= numberofpoints
```

```
if Re(j) > 2100
    F(n) = (-2.*log((e./3.7065.*PD) - (5.0452./Re(j)).* ...
        log(((1/2.8257).*((e./PD)^1.1098))+ ...
        (5.8506./(Re(j).^(.8981)))))).^(-2);
else
    F(n) = 64./Re(j);
end

P_turbinein(n) = -(nB_density*g).*(((F(n).*(Leq/PD)).*((Vbt(n).^2)./(2*g)))...
    - (P_boilerout(n)./(nB_density*g)));

i = i+1;
j = j+1;
n = n+1;
end
Reynolds_Number1 = transpose(Re);
Friction_Factor1 = transpose(F);
Pressure_Turbine_in = transpose(P_turbinein);

Pr = kinematic_viscosity / k_nbutane;
Nu = 0.023.*(Reynolds_Number1.^0.8).*(Pr^0.3);
h1a = (Nu.*k_nbutane)./PD;
critical = k_nbutane./h1a
R_conv1 = 1./(pi.*PD.*Leq.*h1a);
R_conduction1 = log((PD+(2*thickness))/PD) ./ (2*pi*Leq*k_bearsteel);

R_conv2 = 1 ./ (pi*PD*Leq)

%%%%%% Turbine To Compressor %%%%%%
P_lossturbine = 500000;
P_turbineout = P_turbinein - P_lossturbine;
transpose(P_turbineout);
F2 = zeros(1,numberofpoints); %rows x columns
P_compressin = zeros(1,numberofpoints);
Leq2 = 25; % [m]
Vtc =  linspace(1,6,numberofpoints) ; %[m/s] %Velocity From boiler to inlet of Turbine
Re2 = (Vtc.*PD ./ kinematic_viscosity);

i = 1;
j = 1;
n = 1;

%Finding the Frictional Factor for different Re #
while i <= numberofpoints

if Re2(j) > 2100
    F2(n) = (-2.*log((e./3.7065.*PD) - (5.0452./Re2(j)).* ...
        log(((1/2.8257).*((e./PD)^1.1098))+ ...
        (5.8506./(Re2(j).^(.8981)))))).^(-2);
else
    F2(n) = 64./Re2(j);
end

P_compressin(n) = -(nB_density*g).*((F2(n).*(Leq2/PD)).*((Vtc(n).^2)./(2*g))...
    - (P_turbineout(n)./(nB_density*g)));

i = i+1;
j = j+1;
n = n+1;
end
Pressure_Compressor_in = transpose(P_compressin)
```

```
%%%%%% Compressor To Boiler %%%%%%

P_compressout(1:numberofpoints) = 1100000; % this would be a set value
F3 = zeros(1,numberofpoints); %rows x columns
P_boilerin = zeros(1,numberofpoints);
Leq3 = 25; % [m]
Vcf =  linspace(1,6,numberofpoints) ; %[m/s] %Velocity From boiler to inlet of Turbine
Re3 = (Vcf.*PD ./ kinematic_viscosity);
i = 1;
j = 1;
n = 1;

while i <= numberofpoints

if Re3(j) > 2100
    F3(n) = (-2.*log((e./3.7065.*PD) - (5.0452./Re3(j)).* ...
        log(((1/2.8257).*((e./PD)^1.1098))+ ...
        (5.8506./(Re3(j).^(.8981)))))).^(-2);
else
    F3(n) = 64./Re3(j);
end

P_boilerin(n) = -(nB_density*g).*((F3(n).*(Leq3/PD)).*((Vcf(n).^2)./(2*g))...
    - (P_compressout(n)./(nB_density*g)));

i = i+1;
j = j+1;
n = n+1;
end
Pressure_Boiler_in = transpose(P_boilerin);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Team 2 Cummins Energy Savings
% Simulation Package

clear all,clc,close all

%% Exhaust Properties
rho_air = 0.7461; % kg/m^3 [1]
Cp_air = 1.0061; % kJ/kg*K
K_air = 1.4019; % Specific Heat Ratio cp/cv
Cv_air = Cp_air / K_air; % kJ/kg*K

prop_air = [ rho_air Cp_air K_air Cv_air];

%% ENGINE SERIES DETAILS

% Engine Power rating values. All values are in Horsepower.
ISB_HP = [ 185 190 205 210 225 240 245 260 275 ];
ISC_HP = [ 225 240 260 285 300 315 330 350 ];
ISL_HP = [ 310 330 ];
ISM_HP = [ 280 310 330 350 370 400 425 450 500 ];
ISX_HP = [ 400 450 475 500 600 ];

% Exhaust temperature values.
% All values are in Farenheit.
ISB_OUT_TEMP = [ 698 801 831 857 892 812 812 886 956];
ISC_OUT_TEMP = [ 706 746 765 833 860 919 927 966 ];
ISL_OUT_TEMP = [ 891 933 ];
ISM_OUT_TEMP = [ 670 721 742 720 737 737 969 789 965 ];
ISX_OUT_TEMP = [ 655 696 842 905 975 ];

% Mass flow rate values.
% All values are in CFM.
ISB_CFM_OUT = [ 1257 1250 1246 1313 1311 1456 1456 1592 1673 ];
ISC_CFM_OUT = [ 1417 1485 1578 1531 1578 1686 1758 1841 ];
ISL_CFM_OUT = [ 1681 1740 ];
ISM_CFM_OUT = [ 1523 1528 1610 1778 1853 1853 2171 2030 2341 ];
ISX_CFM_OUT = [ 2036 2218 2504 2633 3202 ];
% CONVERSION TO VALUES IN kg/s.

ISB_kgs_OUT = ISB_CFM_OUT*(.0283168)*(1/60) * rho_air;
ISC_kgs_OUT = ISC_CFM_OUT*(.0283168)*(1/60) * rho_air;
ISL_kgs_OUT = ISL_CFM_OUT*(.0283168)*(1/60) * rho_air;
ISM_kgs_OUT = ISM_CFM_OUT*(.0283168)*(1/60) * rho_air;
ISX_kgs_OUT = ISX_CFM_OUT*(.0283168)*(1/60) * rho_air;

ISB_avg_mdot = mean(ISB_kgs_OUT);
ISC_avg_mdot = mean(ISC_kgs_OUT);
ISL_avg_mdot = mean(ISL_kgs_OUT);
ISM_avg_mdot = mean(ISM_kgs_OUT);
ISX_avg_mdot = mean(ISX_kgs_OUT);
EngineAverages = [ISB_avg_mdot, ISC_avg_mdot, ISL_avg_mdot,ISM_avg_mdot,
ISX_avg_mdot];
%% Types of engines and mass flow
massFlowOut = [ ];
i = 1;
j = 1;
k = 1;
NumOfEngines = 0;
EngineSeries = 0;
NumEng = zeros(1,5);

while EngineSeries ~= 6
    disp('Please choose 1 = ISB,2 = ISC,3 = ISL,4 = ISM,5 = ISX,6 = DONE')
    EngineSeries = input('Which engine series is this?');
    % Entries allowed: ISB,ISC,ISL,ISM,ISX.

    if EngineSeries == 1
        massflow = ISB_kgs_OUT;
        i = 1;
    elseif EngineSeries == 2
        massflow = ISC_kgs_OUT;
        i = 2;
    elseif EngineSeries == 3
        massflow = ISL_kgs_OUT;
        i = 3;
    elseif EngineSeries == 4
        massflow = ISM_kgs_OUT;
        i = 4;
    elseif EngineSeries == 5
        massflow = ISX_kgs_OUT;
        i = 5;
    elseif EngineSeries == 6
        break;
    else
        error('Please select a value between 1-6.')
    end

   NumOfEngines = input('How many of these engines are running currently?');
   NumEng(i) = NumOfEngines;

CombinedMassFlow(i) = EngineAverages(i)*NumEng(i); % Store the number of engine series
into combined mass flow array. The sum of this array is the total mass flow.
end % END WHILE
```

```
disp(' The mass flow rate [kg/s] for different types of engines in the series are:')
TotalMassFlow = sum(CombinedMassFlow)

%% Temperature Exhaust

T_in = 400; % Degree (F) {typically 400F}
T_C = (T_in - 32)* (5/9); % Output Degree Celsius
T_K = T_C + 273.15; % Output convert to Kelvin.
Tout_exhaust = 212;% [Degree F] (corresponds to 100C, picked from as design parameter
that determines size of heat exchanger.
Tout_exhaust = ((Tout_exhaust-32)*(5/9))  + 273.15; % [K]

%% Working Fluid Properties - Butane
% DENSITY
rho_but = 2.48; % [kg/m^3]
% ABSOLUTE VISCOSITY
visc_but = 7*10^(-6); % [Pa*s]
% SPECIFIC HEAT
Cp_but=1.675; %[kJ/kg*K]
Cp_but_turb=2.2; %[kJ/kg*K]
gamma = 1.096; % Specific heat ratios
% THERMAL CONDUCTIVITY
k_but = 1.70*10^(-2); %[W/m*K]
% PRANDTL NUMBER
Pr_but = 0.69; % [UNITLESS]
% MASS FLOW RATE
mdot_but =  7.00; % [kg/s]
% RUN Solar_Thermal_Collectors function. (Daily)
Day = input('What day of the year is it? (1-365)');
Time = input('What time is it?');
T_Collector_out = Solar_Thermal_Collectors(Day,Time);

% TEMPERATURE INLET
Tin_but = T_Collector_out; % [K]
% Run code from solar collectors.
% TEMPERATURE OUTLET
Tout_but = 145 + 273; % [K]

%% Heat Generated From Exhaust Gases
Q_exhaust = TotalMassFlow*prop_air(2)*(T_K - Tout_exhaust) % Units [kW]
%% Heat Available After Exchange
Q_available = Q_exhaust*.19
T_out_heatexchange = (Q_available/(mdot_but*Cp_but))+Tin_but
T_out_C = T_out_heatexchange - 273.15
%% Turbine Code
Poweroutput = 0.5*Q_available;
T_turbine_in = T_out_heatexchange;
P_turbine_in = 1000; %call from pipe loss code kPa
P_turbine_out = 500; %kPa
T_turbine_out = T_turbine_in .* ((P_turbine_out./P_turbine_in) .^(1-(1/gamma)));
dh = Cp_but_turb*(T_turbine_in-T_turbine_out) % Units: [kJ/kg*K]

True_powerout = mdot_but*dh
Poweraccuracy = True_powerout/Poweroutput *100
%% Compressor Code
T_compress_in = T_turbine_out;
P_compress_out = 1100; %kPa
P_compress_in = P_turbine_out; %%% WE can call from pipe loss code.
T_compress_out = T_compress_in.* ((P_compress_out ./ P_compress_in) .^(1-(1/gamma)));


%% Annual Energy
SystemEfficiencyGuess = (Poweroutput/Q_exhaust)*100 % Percent
```

```
SystemEfficiencyTrue = (True_powerout/Q_exhaust)*100 % Percent
Annual_Energy = True_powerout*3600*runningtime*365; % Units [kJ/day]
Annual_Consumption = 7.6*10^(11); % Units [kJ/year]
Annual_Energy_Savings = (Annual_Energy/Annual_Consumption)*100; % Percent
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Solar Thermal Collectors
% Senior Design Project
function [T_Collector_out] = Solar_Thermal_Collectors(day,time)


numberofpoints = 24;
SRT = xlsread('Solar_Power','Sheet1','G11:G375'); % Sunrise Time
SST = xlsread('Solar_Power','Sheet1','H11:H375'); % Sunset Time

%day = input('What day of the year is it = ');
time_rise_solar = SRT(day);
time_set_solar = SST(day);
solar_time = linspace(time_rise_solar,time_set_solar,numberofpoints);
solar_angle = zeros(1,numberofpoints);
zenith = zeros(1,numberofpoints);
alpha = zeros(1,numberofpoints);
as = zeros(1,numberofpoints);
cosi = zeros(1,numberofpoints);


%%% Coordinates for Indiana %%%
I_o = 1367; %W/m^2
a = 0.14;
h = .192; % km above sea level
AM = 1.5; % Air Mass Index [100 units]
aw = 0;
rho = 0.4;


lamda = Coordinate_Time(39,12,4.49298); % lattitude
W = Coordinate_Time(85,54,13.0428); % Lonigtude
beta = 0;


%%% Average Insolation Equation %%%
I_local = I_o.*((1-a.*h).*(0.7.^(((AM).^0.678))) + (a.*h));
declination = 23.44.*sind(360*((day-80)/365.25));

i = 1;
j = 1;

while i <= numberofpoints
    solar_angle(i) = (360/24).*(solar_time(i) - 12);
    zenith(i) = acosd(sind(declination).*sind(lamda) + ...
        cosd(declination).*cosd(lamda).*cosd(solar_angle(i)));

    alpha(i) = (90 - zenith(i));
    as(i) = sind(cosd(declination).*sind(solar_angle(i))./cosd(alpha(i)));


    if lamda > declination
        as(j) = 180 - abs(as(i));
    else
        as(j) = as(i);
    end

    cosi(i) = cosd(alpha(i)).*cosd(as(i)).*sind(beta) + sind(alpha(i)).*cosd(beta);
    j = j+1;
    i = i+1;
end

Direct_Beam = I_local.*cosi; % Regardless of what we use this is correct (Horizontal
equations and elevated equations)
z = 1;
%{
while z <= numberofpoints
    if Direct_Beam(z) < 0
        Direct_Beam(z) = 0;
    else
        Direct_Beam(z) = Direct_Beam(z);
    end
    z = z+1;
end
%}
if (day < 32)
    C = 0.048;
    Temperature_ambient = 2 + 273;
elseif (day < 59) && (day >= 32)
    C = 0.050;
    Temperature_ambient(1:numberofpoints) = 4.556 + 273;
elseif (day < 90) && (day >=59)
    C = 0.061;
    Temperature_ambient(1:numberofpoints) = 10.944 + 273;
elseif (day < 120) && (day >=90)
    C = 0.087;
    Temperature_ambient(1:numberofpoints) = 17.444 + 273;
elseif (day < 151) && ( day >= 120)
    C = .111;
    Temperature_ambient(1:numberofpoints) = 22.667 + 273;
elseif (day < 181) && (day >= 151)
    C = .124;
    Temperature_ambient(1:numberofpoints) =  27.722 + 273;
elseif (day < 212) && (day >=181)
    C = .126;
    Temperature_ambient(1:numberofpoints) = 29.444 + 273;
elseif (day < 243) && (day >= 212)
    C = .112;
    Temperature_ambient(1:numberofpoints) = 28.889 + 273;
elseif (day < 273) && (day >= 243)
    C = .082;
    Temperature_ambient(1:numberofpoints) = 25.333 + 273;
elseif (day < 304) && (day >= 273)
    C = 0.063;
    Temperature_ambient(1:numberofpoints) = 18.333 + 273;
elseif (day < 334) && (day >= 304)
    C = 0.053;
    Temperature_ambient(1:numberofpoints) = 11.111 + 273;
else
    C = 0.057;
    Temperature_ambient(1:numberofpoints) = 3.333 + 273;
end

Diffuse_Beam = C.*I_local;
Total_Beam = Direct_Beam + Diffuse_Beam;


%% Collector Energy Balance

massflownbutane(1:numberofpoints) = 7; %kg/s
```

```
cp_nbutane = 1900; %J/kg*K
Collector_area = 2100; %m^2 CTC Roof
Solar_transmittance = 0.92; %AE Collector
Solar_Absorptance = 0.96; %AE Collector
Collector_storage = 0; %J
Global_Conductance = 8; %W/m^2*K
Insolation = Total_Beam * Solar_transmittance;     %correct %W/m^2
G = massflownbutane / Collector_area;
T_f_in = 80 + 273.15;
T_f_out = T_f_in + (50 .* C ./ .126) .* (Insolation ./ max(Insolation));
F_r = (G * cp_nbutane  .*( T_f_out - T_f_in )) ./ (( Solar_Absorptance .* Insolation)
- Global_Conductance * ( T_f_in - Temperature_ambient ));
heat_Useful = F_r .* Insolation .* Solar_Absorptance .* Collector_area -
(Global_Conductance .* (T_f_in - Temperature_ambient) .* F_r .* Collector_area);
%Heat_Useful = abs(heat_Useful)
%Heat_lost = (Global_Conductance .* Collector_area .* (T_f_out + 273.15 -
Temperature_ambient))
Heat_lost = (Insolation .* Collector_area) - heat_Useful;
Temperaturedifference = heat_Useful ./ (massflownbutane.*cp_nbutane);

T_Collector_out = Temperaturedifference + T_f_in; %K
T_Collector_out_C = T_Collector_out - 273.15 %C

TotalTempchange = sum(Temperaturedifference); % Not used.

Heat_total = heat_Useful + Heat_lost;

Collector_efficiency = heat_Useful ./ Heat_total; % Should be 30-40%
%{
while tcounter<=48
    if tcounter/2 < time_rise_solar || tcounter/2 > time_set_solar
        t_Collector_out(tcounter) = T_f_in;
    else
        t_Collector_out(tcounter) = t_Collector_out(tcounter);
    end
    tcounter = tcounter + 1;
end
%}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%

% Solar Thermal Collectors year
% Senior Design Project

clear all
clc

numberofpoints = 20;
SRT = xlsread('Solar_Power','Sheet1','G11:G375'); % Sunrise Time
SST = xlsread('Solar_Power','Sheet1','H11:H375'); % Sunset Time

solar_angle = zeros(365,numberofpoints);
zenith = zeros(365,numberofpoints);
alpha = zeros(365,numberofpoints);
as = zeros(365,numberofpoints);
cosi = zeros(365,numberofpoints);
solar_time_matrix = zeros(365,numberofpoints); % Size 365x20
Direct_Beam = zeros(365,numberofpoints);


lamda = Coordinate_Time(39,12,4.49298); % lattitude
W = Coordinate_Time(85,54,13.0428); % Lonigtude
beta = 0;
```

```
I_o = 1367; %W/m^2
a = 0.14;
h = .192; % km above sea level
AM = 1.5; % Air Mass Index [100 units]
aw = 0;
rho = 0.4;
I_local = I_o.*((1-a.*h).*(0.7.^(((AM).^0.678))) + (a.*h));
i = 1;
j = 1;
for day = 1:365

    time_rise_solar = SRT(day); % Reads the sunrise time, assigns here.
    time_set_solar = SST(day); % Reads the sunset time, assigns here.
    solar_time = linspace(time_rise_solar,time_set_solar,numberofpoints)';
    declination = 23.44.*sind(360*((day-80)/365.25));

while i <= numberofpoints % While i runs through 20 evenly spaced columns
    solar_time_matrix(j,i) = solar_time(i);    % This should assign the solar times
for each day.
    solar_angle(j,i) = (360/24).*(solar_time_matrix(j,i) - 12);

    zenith(j,i) = acosd(sind(declination).*sind(lamda) + ...
        cosd(declination).*cosd(lamda).*cosd(solar_angle(j,i)));
    alpha(j,i) = (90 - zenith(j,i));
    as(j,i) = sind(cosd(declination).*sind(solar_angle(j,i))./cosd(alpha(j,i)));

    if lamda > declination
        as(j,i) = 180 - abs(as(j,i));
    end % END IF STATEMENT

 cosi(j,i) = cosd(alpha(j,i)).*cosd(as(j,i)).*sind(beta) +
sind(alpha(j,i)).*cosd(beta);
 Direct_Beam(j,i) = I_local.*cosi(j,i);

 %% CLOUDINESS FACTOR
 if (day < 32)
    C = 0.048;
    Temperature_ambient = 2 + 273;
elseif (day < 59) && (day >= 32)
    C = 0.050;
    Temperature_ambient(1:numberofpoints) = 4.556 + 273;
elseif (day < 90) && (day >=59)
    C = 0.061;
    Temperature_ambient(1:numberofpoints) = 10.944 + 273;
elseif (day < 120) && (day >=90)
    C = 0.087;
    Temperature_ambient(1:numberofpoints) = 17.444 + 273;
elseif (day < 151) && ( day >= 120)
    C = .111;
    Temperature_ambient(1:numberofpoints) = 22.667 + 273;
elseif (day < 181) && (day >= 151)
    C = .124;
    Temperature_ambient(1:numberofpoints) =  27.722 + 273;
elseif (day < 212) && (day >=181)
    C = .126;
    Temperature_ambient(1:numberofpoints) = 29.444 + 273;
elseif (day < 243) && (day >= 212)
    C = .112;
    Temperature_ambient(1:numberofpoints) = 28.889 + 273;
elseif (day < 273) && (day >= 243)
    C = .082;
    Temperature_ambient(1:numberofpoints) = 25.333 + 273;
elseif (day < 304) && (day >= 273)
```

```
    C = 0.063;
    Temperature_ambient(1:numberofpoints) = 18.333 + 273;
elseif (day < 334) && (day >= 304)
    C = 0.053;
    Temperature_ambient(1:numberofpoints) = 11.111 + 273;
else
    C = 0.057;
    Temperature_ambient(1:numberofpoints) = 3.333 + 273;
end

Diffuse_Beam(j,i) = C.*I_local.*(1 + cosd(beta)) ./ 2;
Reflected_Beam(j,i) = rho.*I_local.*sind(alpha(j,i) + C).*(sind(beta/2)^2);
Total_Beam(j,i) = Direct_Beam(j,i) + Diffuse_Beam(j,i) + Reflected_Beam(j,i);


 i = i+1;

end % END WHILE LOOP
    i = 1;
    j = j+1;


end
   solar_time_matrix;
   solar_angle;
   zenith;
   alpha;
   as;
   cosi;
   Direct_Beam;
   Diffuse_Beam
   Total_Beam
```