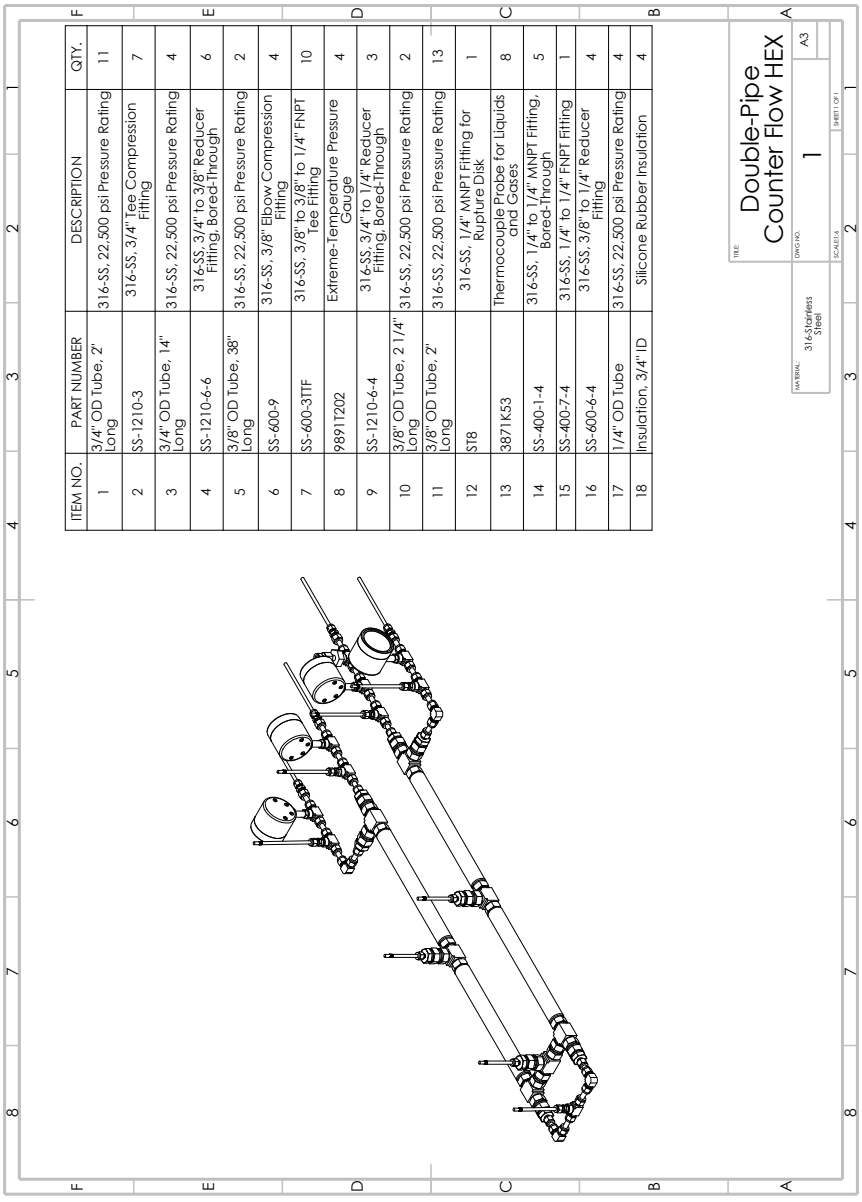
Appendix A

Heat Exchanger Drawing and Bill of Materials



ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1	3/4" OD Tube, 2' Long	316-SS, 22,500 psi Pressure Rating	11
2	SS-1210-3	316-SS, 3/4" Tee Compression Fitting	7
3	3/4" OD Tube, 14" Long	316-SS, 22,500 psi Pressure Rating	4
4	SS-1210-6-6	316-SS, 3/4" to 3/8" Reducer Fitting, Bored-Through	6
5	3/8" OD Tube, 38" Long	316-SS, 22,500 psi Pressure Rating	2
6	SS-600-9	316-SS, 3/8" Elbow Compression Fitting	4
7	SS-600-3TTF	316-SS, 3/8" to 3/8" to 1/4" FNPT Tee Fitting	10
8	9891T202	Extreme-Temperature Pressure Gauge	4
9	SS-1210-6-4	316-SS, 3/4" to 1/4" Reducer Fitting, Bored-Through	3
10	3/8" OD Tube, 2 1/4" Long	316-SS, 22,500 psi Pressure Rating	2
11	3/8" OD Tube, 2' Long	316-SS, 22,500 psi Pressure Rating	13
12	ST8	316-SS, 1/4" MNPT Fitting for Rupture Disk	1
13	3871K53	Thermocouple Probe for Liquids and Gases	8
14	SS-400-1-4	316-SS, 1/4" to 1/4" MNPT Fitting, Bored-Through	5
15	SS-400-7-4	316-SS, 1/4" to 1/4" FNPT Fitting	1
16	SS-600-6-4	316-SS, 3/8" to 1/4" Reducer Fitting	4
17	1/4" OD Tube	316-SS, 22,500 psi Pressure Rating	4
18	Insulation, 3/4" ID	Silicone Rubber Insulation	4

Figure A.1: Drawing with Bill of Materials for Experimental Heat Exchanger

Appendix B

MATLAB Code

```
function [cyc_efficiency2,net_work,energy,...
    p1,T2,p2,T3,p3,T4,p4,...
    p5,p6] = BraytonCycle(m_dot,p1,p_a,PR_c,UA,...
    fluid,mode,m_SC,m_CT)

% Inputs:
% m_dot: the mass flow in the cycle [kg/s]
% p1: flow pressure at inlet of the compressor [kPa]
% p_a: ambient air pressure [kPa]
% PR_c: pressure ratio of the compressor
% UA: conductance of recuperator [W/K]
% fluid: working fluid for the system
% Mode: 1(constant property model), 2(use of FIT),3(use of
   REFPROP),
%       or property tables for interpolation
% graph: 1(on), 0 (off)
% m_SC: mass flow rate of PTSC [kg/s]
% m_CT: mass flow rate of water in cooling tower [kg/s]

% outputs:
% cyc_efficiency: total cycle efficiency [1st law efficiency, 2
   nd law
%                 efficiency]
% net_work: net power output from the cycle [W]
% energy: check to see if energy is conserved (should be zero)
   - does not
%         account for pressure drops


```

```
27
28 [p2,p3,p4,p6,p5,~] = findPressures(p1,PR_c);
29
30 [T_H_in,SS,Q_solar] = Solar_Code2(m_SC);
31 table1 = readtable('Dry_Bulb_Temp.xlsx');
32 Humidity = table1.Humidity;
33 T_a_in = table1.DRY_BULB_TEMP;
34 t = table1.TIME;
35 T = ones(length(SS),1);
36 leng = length(T);
37
38 T1 = (300)*T;
39 T2 = T;
40 T3 = T;
41 T4 = T;
42 T5 = T;
43 T6 = T;
44 energy = T;
45 q_comb = T;
46 N = T;
47 Power_c = T;
48 Power_T = T;
49 q_HEX = T;
50 Q_CT = T;
51 p1 = p1*T;
52 p2 = p2*T;
53 p3 = p3*T;
54 p4 = p4*T;
55 p5 = p5*T;
56 net_work = T;
57 cyc_efficiency =T;
58 cyc_efficiency2 = T;
59 n_c = T;
60 n_T = T;
61 NTU = T;
62 for const = 1:(leng)
63         % solve for state after compressor
64         [T2(const),~,N(const),Power_c(const),n_c(const),~] =
              Compressor(m_dot,T1(const),p1(1),p2(1),fluid,mode);
65
66         % solve for recuperator outlets
```

```
67          [T3(const),NTU(const)] = HEX_bettersolve(T_H_in(const),
                T2(const),p3(1),p4(1),p3(1),p4(1),m_dot,m_SC,UA,
                fluid,'Water',mode);
68
69         %Combustor adds heat to cycle
70         [T4(const),q_comb(const)] = Combustor(T3(const),m_dot,p4
              (1),fluid,SS);
71          % solve for state after turbine
72         [T5(const),Power_T(const),~,n_T(const)] = Turbine(T4(
              const),p4(const),p5(1),fluid,mode,N(const),m_dot);
73          % Cooling Tower
74
75          [T6(const),~,T_W(const)] = CoolingTower(m_CT,30,
                Humidity(const),T_a_in(const));
76
77          [T1(const+1)] = RadiatorCT(m_CT,m_dot,p5(1),p_a(1),T5(
                const),T6(const));
78
79          net_work(const) = Power_T(const)-Power_c(const);
80          cyc_efficiency(const) = net_work(const)./(q_comb(const)
                -Q_solar(const));
81          cyc_efficiency2(const) = cyc_efficiency(const).*100;
82          energy(const) = (q_comb(const)+Q_CT(const)) - net_work(
                const);
83          T_R = T4(const)/T1(const);
84
85
86  end
87
88          T1_C = T1-273;
89          T1_C = T1_C(2:end,1);
90
91          T2_C = T2-273;
92
93          T3_C = T3-273;
94
95          T4_C = T4-273;
96
97          T5_C = T5-273;
98
99          T6_C = T6-273;
100
```

```
        T_SC_C = T_H_in-273;


 filename = 'DATARESULTS0414MDOT.xlsx';
 writematrix(T1_C,filename,'Sheet','T_comp_in');
 writematrix(T2_C,filename,'Sheet','T_HEX_in');
 writematrix(T3_C,filename,'Sheet','T_comb_in');
 writematrix(T4_C,filename,'Sheet','T_Turb_in');
 writematrix(T5_C,filename,'Sheet','T_rad_in');
 writematrix(T6_C,filename,'Sheet','T_CT_out');
 writematrix(cyc_efficiency2,filename,'Sheet','Therm_eff');
 writematrix(Q_solar,filename,'Sheet','Q_Solar');
 writematrix(q_comb,filename,'Sheet','Q_comb');
 writematrix(T_SC_C,filename,'Sheet','T_H_in');
 writematrix(n_T,filename,'Sheet','Turb_eff');
 writematrix(n_c,filename,'Sheet','Comp_eff');
 writematrix(NTU,filename,'Sheet','NTU');




% plot(za(6),t,SS);
% xlabel(za(6),'Time, hrs');
% ylabel(za(6),'Solar Irradiance, W/m^2');
% print(zf(6),'-dpng','TimevsSS.png');


plot(za(7),time,T3_C);
xticks(za(7),[0 1 2 3 4 5 6 7 8 9 10 11 12]);
xticklabels(za(7),{'January','February','March','April','May','
   June','July','August','September','October','November','
   December', 'January'});
ylabel(za(7),'Compressor Inlet Temp., ^oC');
%

end
```

```
% This code solves for the parabolic trough solar collector using equations
% found in Erdogan, A., Kizilkan, O., & Colpan, C. O. (2019). Thermodynamic
% analysis of a supercritical closed Brayton cycle integrated with
% parabolic trough solar collectors. Journal of Thermal Analysis and
% Calorimetry. doi:10.1007/s10973-019-09199-0
%By Kenneth Weddle, 3/23/2022

function [T_H_in,SS,q_solar] = Solar_Code2(m_HTF)
%Inputs:
clc;
format longg;

table1 = readtable('SSDATA.xlsx');
time = table1.TIME_T_HOURS;
% SS = table1.SOLAR_INTENSITY_SS;
SS = table1.GHI;
% SS = table1.DHI;
T_r = 40+273.15; %receiver temp, K
T_PTSC_in = 40+273.15;
L = 35; %PTSC Length, m
w = 4.8;    %PTSC aperture width, m
D_o =  0.09;  %Outside diameter of pipe receiver, m
D_i =  0.08; %Inside diameter of pipe receiver, m
D_cover =  0.15;    %Diameter of glass cover, m
k_tube = 25/1000 ;    %Conductivity of PTSC tube, kW/m*K
e_receiver = 0.92;   %Emissivity of receiver
e_cover = .87 ;      %Emissivity of glass cover
p = 0.96   ;     %Reflectivity of the aperture surface
tau = 0.94  ;    %Transmissivity of receiver
alpha = 0.93 ;      %Absoptivity of glass cover
K = 0.75 ;       %Incidence angle modifier
T_a = 20+273.15;    %Temperature of the environment, K
density =   965;      %Density of thermal oil, kg/m^3
mu = 0.000642         ;  %Dynamic viscosity, kg/m*s
T_g = 25.001+273.15 ;   %Temperature of the Glass Cover, K
sigma = 5.6704*(10^-8);   %Stefan-Boltzmann Constant, W/m^2*K^4
A_r = 2*pi*(D_i/2)*L;       %Area of Receiver, m^2
A_g = 2*pi*(D_cover/2)*L;   %Area of Glass Cover, m^2
```

```
c_p_HTF = 4.1592;          %Specific heat of thermal oil, kJ/kg*K
A_a = L*w;            %Area of Apeture, m^2
V_oil = (m_HTF*4)/(density*pi*(D_i^2));       %Velocity, m/s
cond = 0.64574/1000    ;      %Liquid Thermal Conductivity of Oil, kW/m*K
Pr = (mu*(c_p_HTF))/cond;
Re = (density*V_oil*D_i)/mu;
%%
if Re <= 2300
    Nu = 4.634;
else
    Nu = 0.023*(Re.^.8)*(Pr^0.4);
end
%%
h_fi = (Nu*k_tube)/D_i ;         %Radiative Heat Trans. Coeff. of inside receiver, kW/m^2*K
h_r = e_receiver*sigma*((T_g^4 - T_a^4)/(T_g-T_a)) ;        %kW/m^2*K
h_r = h_r./1000;
T_glass = (A_r*h_r*T_r+(A_g*(h_r+h_fi)*T_a))/(A_r*h_r+(A_g*(h_r+h_fi)));
T_diff = abs(T_glass - T_g); %#ok<NASGU>
if T_diff <= 0.001
    T_g = T_glass;
end
if T_g > T_glass

    T_g = T_glass+T_diff;
else
    T_g = T_glass-T_diff;
end
%%
U_L = (A_r./((h_fi+h_r)*A_g)).^-1 ; %Heat Loss Coefficient for Solar Collector, kW/m^2*K
%%
F_prime = (1./U_L)./((1./U_L)+(D_o./(h_fi.*D_i))+((D_o./(2*k_tube))+log(D_o./D_i)));

%%
A = A_a-A_r;
%%
```

```
74 F_R = ((m_HTF.*c_p_HTF)/(A_r.*U_L)).*(1-exp(-((U_L.*F_prime*A_r
   )/(m_HTF.*c_p_HTF))));
75 %%
76 q_dot1 = F_R*((SS/1000)*p*tau*alpha*K*(A)*U_L*(T_PTSC_in-T_a));
         %Useful Heat Gain, kW
77 q_solar = q_dot1*1000;                    %Useful Heat Gain, W
78 %%
79 T_H_in = T_PTSC_in + (q_dot1./(m_HTF*c_p_HTF));
80
81 end
```

```
function [T_C_out,NTU] = HEX_bettersolve(T_H_in,T_C_in,p_H_in,
   p_H_out,p_C_in,p_C_out,m_dot_H,m_dot_C,UA,fluid_C,fluid_H,
   mode)
%
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% description: discretized e-NTU method with HEX's in series
% development notes: John Dyerby's thesis and Nellis and Klein
% section 8.6.3 extension

% Inputs:
% T_H_in: inlet temperature at hot side of HEX [K]
% T_C_in: inlet temperature at cold side of HEX [K]
% p_H_in: hot side inlet pressure [kPa]
% p_H_out: hot side outlet pressure [kPa]
% p_C_in: cold side inlet pressure [kPa]
% p_C_out: cold side outlet pressure [kPa]
% m_dot_H: hot side mass flow rate [kg/s]
% m_dot_C: cold side mass flow rate [kg/s]
% UA: conductance [W/K]
% fluid_C: cold side fluid
% fluid_H: hot side fluid
% Mode: 1(constant property model), 2(use of FIT),3(use of
   REFPROP),
%       or property tables for interpolation


% Outputs:
% T_C_out: outlet temperature at cold side of HEX [K]
% q_dot_HEX: heat transfer rate of HEX [W]
% h_C_out: outlet enthalpy of cold side fluid [J/kg*K]



discretize HEX
    N = 1;                      % number of sub HEX's


calculate pressures for points along HEX
preallocate space
    p_H = zeros(1,(N+1));
```

```
38      p_C = zeros(1,(N+1));
39
40  % initial pressures values for inlets of HEX
41      p_H = p_H_in;                % hot side inlet pressure
42      p_C = p_C_out;               % cold side inlet pressure
43
44  % find pressure change for each sub HEX
45      ploss_C = (p_C_in-p_C_out)/N;
46      ploss_H = (p_H_in-p_H_out)/N;
47
48  % % enthalpy values for outlet and inlet of each sub HEX
49      for i = 2:(N+1)
50          p_H(i) = p_H(i-1)-ploss_H;            % calculate hot
               side pressure at outlet of sub HEX
51          p_C(i) = p_C(i-1)+ploss_C;            % calculate cold
               side pressure at outlet of sub HEX
52      end
53  find bounds for fzero
54      [ Tmin, Tmax ] = boundFind(T_H_in,T_C_in,p_H,p_C,m_dot_H,
           m_dot_C,UA,fluid_C,fluid_H,mode,N );
55  %
56          if isnan(Tmin) || isnan(Tmax)
57      % if boundFind failed to find a valid interval, end the
           function with
58      % NaN outputs
59              T_H_out = NaN;
60              T_C_out = NaN;
61              h_C_out = NaN;
62              p_H = NaN;
63              p_C = NaN;
64              T_H = NaN;
65              T_C = NaN;
66          else
67  %       find zero to obtain T_H_out
68  %               options = optimset('TolX',1E-8);              %
       set tolerancing on solver step size
69              options=[];                                  % sets
                   tolerancing to defualt
70              [T_H_out,~] = fzero(@errorGen,[Tmin,Tmax],options,
                   T_H_in,T_C_in,p_H,p_C,m_dot_H,m_dot_C,UA,fluid_C
                   ,fluid_H,mode,N);
71
```

```
72      eps = .85;                  %Desired Effectiveness of HEX
73      %%%%%%%%%%%%%%%%%%%%%% only to find T_C_out
           %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
74      % find enthalpies of fully specified states
75              [~,~,h_H_in] = getPropsTP(T_H_in,p_H(1),fluid_H,
                   mode,1);      % hot side inlet properties
76              [~,~,h_H_out] = getPropsTP(T_H_out,p_H(1),fluid_H,
                   mode,1);    % hot side outlet properties
77              [~,~,h_C_in] = getPropsTP(T_C_in,p_C(1),fluid_C,
                   mode,1);      % cold side inlet properties
78
79      % calculate total heat transfer
80              q_dot_HEX = m_dot_H*eps*(h_H_in-h_H_out); % total
                   heat transfer rate
81  %     q_dot1 = q_dot_HEX/1000;
82      % calculate enthalpy of cold side outlet
83              h_C_out = h_C_in+q_dot_HEX/m_dot_C;   % specific
                   heat capacity for cold side outlet
84
85      % find temperature of cold side
86              [T_C_out,~,~] = getPropsPH(p_C(1),h_C_out,fluid_C,
                   mode,1); % find outlet temperature for cold side
87
88
89              C_dot_H = m_dot_H*((h_H_in-h_H_out)/(T_H_in-T_H_out
                   ));  %hot side capacitance rates
90          %                               h_H1-h_H2
               T_H1-T_H2
91              C_dot_C = m_dot_C*((h_C_out-h_C_in)/(T_C_out-T_C_in
                   ));  %hot side capacitance rates
92          %                               h_H1-h_H2
               T_H1-T_H2
93              C_dot_min = min(C_dot_C,C_dot_H);
                                        % minimum capacitance rates
94              C_dot_max = max(C_dot_C,C_dot_H);
                                        % maximum capacitance rates
95
96
97
98              C_R = C_dot_min./C_dot_max;
                                             % capacity ratio of
                   sub HEXs
```

```
99                                                  % find the indicies
                                                       of the C_R
                                                       members that are
                                                       1
100             NTU = (1/(C_R - 1))*log((eps-1)/(eps*C_R-1));
                   %NTU Counter Flow Double Pipe
101
102
103         end
104 end
```

```
function [T_out,q_dot] = Combustor(T_in,mdot,p,fluid)
[~,~,h_in] = getPropsTP(T_in,p,fluid,3,1);
T_out = 1100;
[~, ~, h_out] = getPropsTP(T_out,p,'CO2',3,1);
q_dot = mdot*(h_out-h_in);
end
```

```
% This code solves for the parabolic trough solar collector using equations
% found in Wei, X., Li, N., Peng, J., Cheng, J., Hu, J., & Wang, M. (2017).
% Performance Analyses of Counter-Flow Closed Wet Cooling Towers Based on
% a Simplified Calculation Method. Energies, 10(3), 282.
% doi:10.3390/en10030282
% By Kenneth Weddle, 10/4/2022
function [T_W_out,qdot_CT,T_wetbulb] = CoolingTower(m_dotCT,T_W_in,Humidity,T_a_in)

% Inputs
% m_dotCT: Water mass flow rate, [m/s]
% T_W_in: Water inlet temp, [ C ]
% Humidity: Ambient humidity [%]
% T_a_in: Inlet air temp, [ C ]

% Outputs
% T_W_out: Outlet temp of water entering condenser [ C ]
% qdot_CT: Heat transfer rate of cooling tower, [W]

C_p_W = 4.1968*1000;          %Specific heat of water, J/kg* C
u_W = 3.541*10^-4;            % Dynamic Viscosity of water [kg/m*s]
B_int = .64;                  % A constant which is influenced by the coil's geometry and constant water-properties
B_ext = .452;                 % A constant which depends on the thermal properties of air and on the coilAs geometry
C_p_sat = 3.5878*1000;        % Specific heat of saturated air at constant pressure, kJ/( k g   C )
m_dot_a = 1;                  % Mass flow rate of air [kg/s]



T_wetbulb = (T_a_in.*atan(0.151977*(Humidity+8.313659).^.5)+atan(T_W_in-Humidity)-...
atan(Humidity-1.676331)+0.00391838*(Humidity).^(3/2).*atan(0.023101*Humidity)-4.686035);

T_W_out = T_W_in - ((T_W_in-T_wetbulb)/(((C_p_W*m_dotCT)/(B_ext*C_p_sat*(m_dot_a^.8))+...
```

```
32 ((C_p_W*(m_dotCT^.2)*(u_W^.5))/B_int)+((C_p_W*m_dotCT)/(2*
      C_p_sat*m_dot_a))+.5)));
33
34 e = (T_W_in-T_W_out)./(T_W_in-T_wetbulb);
35 qdot_CT = m_dotCT*C_p_W*(T_W_in-T_W_out) ;                    %
      Heat energy of CT, W
36
37 T_W_out = T_W_out + 273.15;
38
39
40 end
```

```
function [T_CO2_out] = RadiatorCT(m_W,m_CO2,p_CO2,p_w,T_CO2_in,
   T_W_in)

c_p_w = refpropm('Cp','T',T_W_in,'P',p_w,'Water');
c_p_CO2 = refpropm('Cp','T',T_CO2_in,'P',p_CO2,'CO2');

C_H = m_CO2*c_p_CO2;
C_C = m_W*c_p_w;

C_min = min(C_C,C_H);
e = .99;

T_CO2_out = T_CO2_in - (e*(C_min*(T_CO2_in-T_W_in))/C_min);
end
```