ME246 Project 3 Report

Analysis of Hybrid Solar - Fossil Fuel Gas Turbine System

Pol Molinas, Sai Kannan Sampath

April 23, 2018

Section 1: Division of Work

For this project, we used github.com to collaborate the coding in MATLAB for the first time. Pol wrote the code for Task 1, 2, 4, and 5. Sai wrote Task 3 and was responsible for debugging all the code. Sai compiled this high quality report, and Pol inserted the result values from MATLAB. Pol did the flowcharts and Sai decided the equations to be used at various stages of the analysis.

Section 2: Computational Scheme & Results

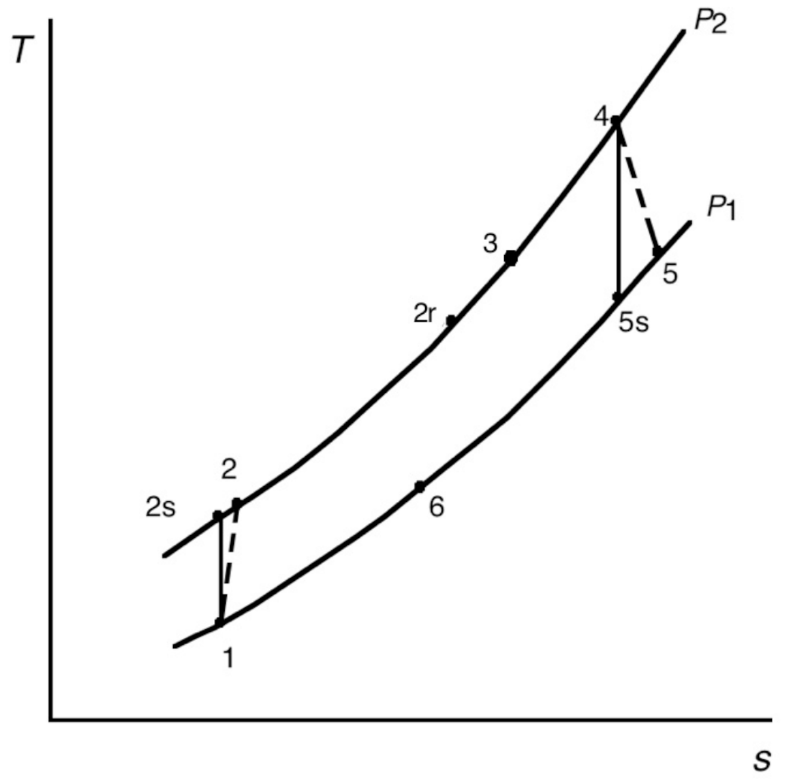

Figure 1: T-S Diagram for Gas Turbine System

Fig. 1 above shows the thermodynamic cycle as a T-S plot for the gas turbine system.

Fig. 2 above shows the physical set-up in a schematic representation for the gas turbine compressor system.

This project deals with analysis of the hybrid solar fossil-fuel gas turbine system in the figures above. The objective of this project was to construct a computer simulation of the

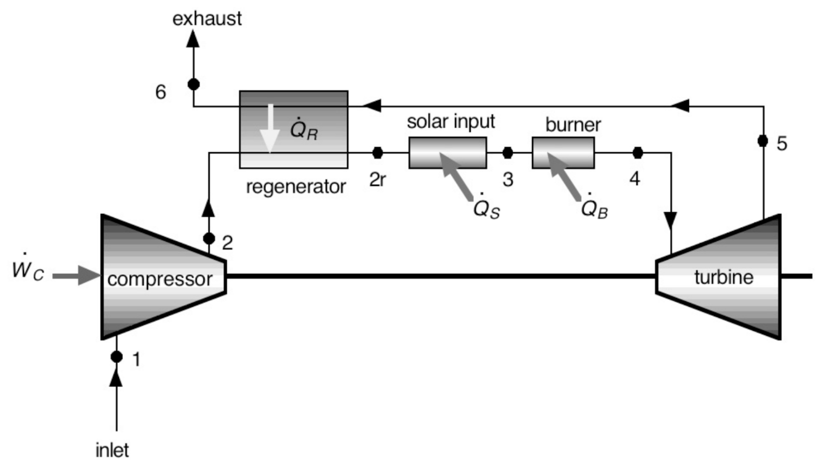

Figure 2: Schematic for Gas Turbine System

performance of the gas turbine system shown in Figures 1 and 2 above. Air at atmospheric pressure $P_1 = 101kPa$ and at a temperature $T_1 = 25°C$ enters the inlet at a flow rate of 6.0 kg/s. Air is treated as an ideal gas with an effective molecular mass $\dot{M}$ of 28.97 kg/k-mol.

Task 1

For this task, we were asked to write a computer program to compute the exit condition from the compressor and the work required per kilogram of air. We first guess a value of compressor outlet temperature $T_{2,guess}$ and then used it to compute the specific heat at constant pressure c_p in kJ/k-mol-K using the average temperature $T_{avg} = 0.5(T_1 + T_2)$ by using the relation for c_p as a function of $T(K)$ for air in the table in figure 3 below.

We then use this value of c_p to calculate c_v and then the ratio of specific heats $k = c_p/c_v$. We then recalculate the value of T_2 from the equation 1 below.

$$\frac{T_2}{T_1} = \frac{P_2}{P_1}^{\frac{k-1}{k}} \tag{1}$$

We iterate this process till we have an error value of $T_2 - T_{2,guess}$ less than 0.2. We then use this value of enthalpy $\hat{h}_2$ at the compressor exit using the below equation 2 where $\hat{h}_1 = 0$.

$$\hat{h}_2 - \hat{h}_1 = c_{p,air}(T_2 - T_1) \tag{2}$$

We then finally calculate the work required per kilogram of air using the equation 3 below.

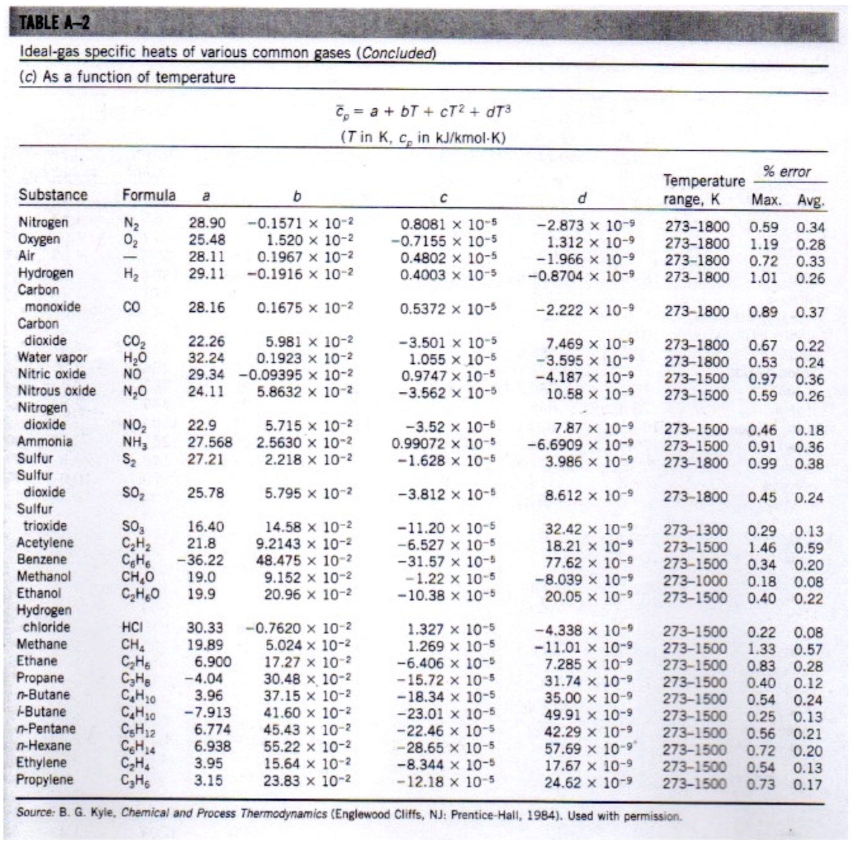

TABLE A–2

Ideal-gas specific heats of various common gases (*Concluded*)

(*c*) As a function of temperature

$$\bar{c}_p = a + bT + cT^2 + dT^3$$

(T in K, c_p in kJ/kmol·K)

Substance	Formula	*a*	*b*	*c*	*d*	Temperature range, K	% error Max.	% error Avg.
Nitrogen	N_2	28.90	-0.1571×10^{-2}	0.8081×10^{-5}	-2.873×10^{-9}	273–1800	0.59	0.34
Oxygen	O_2	25.48	1.520×10^{-2}	-0.7155×10^{-5}	1.312×10^{-9}	273–1800	1.19	0.28
Air	—	28.11	0.1967×10^{-2}	0.4802×10^{-5}	-1.966×10^{-9}	273–1800	0.72	0.33
Hydrogen	H_2	29.11	-0.1916×10^{-2}	0.4003×10^{-5}	-0.8704×10^{-9}	273–1800	1.01	0.26
Carbon monoxide	CO	28.16	0.1675×10^{-2}	0.5372×10^{-5}	-2.222×10^{-9}	273–1800	0.89	0.37
Carbon dioxide	CO_2	22.26	5.981×10^{-2}	-3.501×10^{-5}	7.469×10^{-9}	273–1800	0.67	0.22
Water vapor	H_2O	32.24	0.1923×10^{-2}	1.055×10^{-5}	-3.595×10^{-9}	273–1800	0.53	0.24
Nitric oxide	NO	29.34	-0.09395×10^{-2}	0.9747×10^{-5}	-4.187×10^{-9}	273–1500	0.97	0.36
Nitrous oxide	N_2O	24.11	5.8632×10^{-2}	-3.562×10^{-5}	10.58×10^{-9}	273–1500	0.59	0.26
Nitrogen dioxide	NO_2	22.9	5.715×10^{-2}	-3.52×10^{-5}	7.87×10^{-9}	273–1500	0.46	0.18
Ammonia	NH_3	27.568	2.5630×10^{-2}	0.99072×10^{-5}	-6.6909×10^{-9}	273–1500	0.91	0.36
Sulfur	S_2	27.21	2.218×10^{-2}	-1.628×10^{-5}	3.986×10^{-9}	273–1800	0.99	0.38
Sulfur dioxide	SO_2	25.78	5.795×10^{-2}	-3.812×10^{-5}	8.612×10^{-9}	273–1800	0.45	0.24
Sulfur trioxide	SO_3	16.40	14.58×10^{-2}	-11.20×10^{-5}	32.42×10^{-9}	273–1300	0.29	0.13
Acetylene	C_2H_2	21.8	9.2143×10^{-2}	-6.527×10^{-5}	18.21×10^{-9}	273–1500	1.46	0.59
Benzene	C_6H_6	−36.22	48.475×10^{-2}	-31.57×10^{-5}	77.62×10^{-9}	273–1500	0.34	0.20
Methanol	CH_4O	19.0	9.152×10^{-2}	-1.22×10^{-5}	-8.039×10^{-9}	273–1000	0.18	0.08
Ethanol	C_2H_6O	19.9	20.96×10^{-2}	-10.38×10^{-5}	20.05×10^{-9}	273–1500	0.40	0.22
Hydrogen chloride	HCl	30.33	-0.7620×10^{-2}	1.327×10^{-5}	-4.338×10^{-9}	273–1500	0.22	0.08
Methane	CH_4	19.89	5.024×10^{-2}	1.269×10^{-5}	-11.01×10^{-9}	273–1500	1.33	0.57
Ethane	C_2H_6	6.900	17.27×10^{-2}	-6.406×10^{-5}	7.285×10^{-9}	273–1500	0.83	0.28
Propane	C_3H_8	−4.04	30.48×10^{-2}	-15.72×10^{-5}	31.74×10^{-9}	273–1500	0.40	0.12
n-Butane	C_4H_{10}	3.96	37.15×10^{-2}	-18.34×10^{-5}	35.00×10^{-9}	273–1500	0.54	0.24
i-Butane	C_4H_{10}	−7.913	41.60×10^{-2}	-23.01×10^{-5}	49.91×10^{-9}	273–1500	0.25	0.13
n-Pentane	C_5H_{12}	6.774	45.43×10^{-2}	-22.46×10^{-5}	42.29×10^{-9}	273–1500	0.56	0.21
n-Hexane	C_6H_{14}	6.938	55.22×10^{-2}	-28.65×10^{-5}	57.69×10^{-9}	273–1500	0.72	0.20
Ethylene	C_2H_4	3.95	15.64×10^{-2}	-8.344×10^{-5}	17.67×10^{-9}	273–1500	0.54	0.13
Propylene	C_3H_6	3.15	23.83×10^{-2}	-12.18×10^{-5}	24.62×10^{-9}	273–1500	0.73	0.17

Source: B. G. Kyle, *Chemical and Process Thermodynamics* (Englewood Cliffs, NJ: Prentice-Hall, 1984). Used with permission.

Table A-2 from Thermodynmaics by Y.A. Cengel and M.A. Boles, 7th edition, McGraw Hill, 2011.
Here, the universal gas constant = $\overline{R}$ = 8314 J/kmolK

Figure 3: Table of constants to calculate c_p

$$\dot{W}_{c,act} = \frac{c_{p,air}(T_2 - T_1)}{\eta_{comp}} \tag{3}$$

Appendix shows the flowchart explaining the code used for this task, which was made into a function we called **t2__W__h2__finder**. The first step was to define the constants needed for the calculations, such as initial temperatures and pressures and the constants a,b,c and d to find c_p (refer to the function **findConstants** in Homework 2) Then we guessed a value for T_2 and iterated the equations described above to find a new value of the temperature outlet until the updated value converged by 0.2 degrees.

Task 2

For this task, we were asked to determine the adiabatic flame temperature for three different reactant temperatures (330 K, 550 K and 800 K), $\gamma = 0.25$ and a range of α between 13.09 and 36. We slightly modified our Homework 2 code to achieve this task, looping around different α values and T_r values. The results were then plotted on a log-log plot of $T_{af} - T_r$ versus α. The next step was to use those results and fit a custom curve on the data using the MATLAB fit function:

$$T_{af} - T_r = A \left(\frac{\alpha}{\alpha_{stoich}} \right)^{-n} \tag{4}$$

Finally, we then compared the accuracy of our fitted curve for the three different reactant temperatures.

The results of the adiabatic flame temperature calculations are shown in Table 1. The log-log plot is also shown in Figure 4

Table 1: Adiabatic Flame Temperature as a Function of α for Reactant Temperatures of 300 K, 550K and 800K

α	300K T_{af}	550K T_{af}	800K T_{af}
13.09	2308.3	2503.8	2722.4
16	2038.2	2232.0	2444.0
20	1770.1	1966.0	2176.0
24	1574.9	1774.3	1985.4
28	1425.9	1629.1	1842.1
32	1308.1	1514.9	1730.1
36	1212.7	1422.8	1640.1

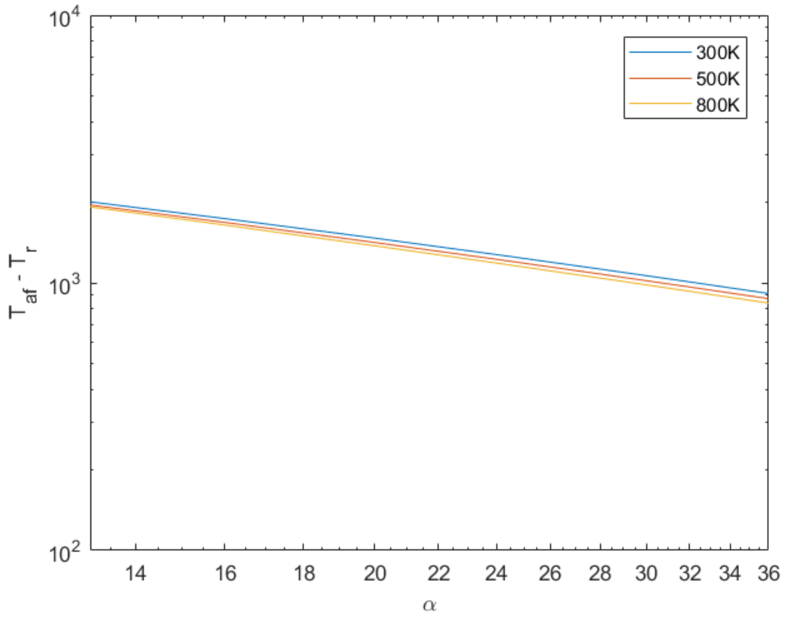

Figure 4: Log-log plot of $T_{af} - T_r$ as a function of α for T_r = 300, 550 and 800 Kelvin

The results of the fitting exercise are show as graphs in Figure 5 and in Table 2.

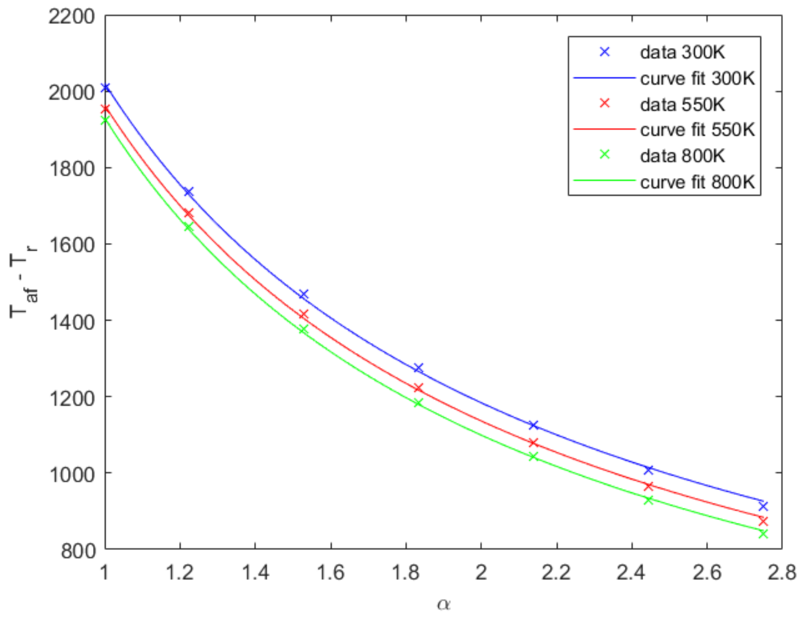

Figure 5: Curve fit of equation (1) on Adiabatic Flame Temperature data as a function of α

Table 2: Curve Fit coefficients with mean and max deviation of three different reactant temperatures

T_r	A	n	Max error (K)	Mean error (K)
300K	1541.8	1.1908	13.6	8.9
550K	1723.5	0.9664	11.2	7.3
800K	1929.9	0.8118	8.8	5.6

Appendix shows the flowchart explaining the code used for this task. The first step of the code was to loop the script from Homework 2 for different values alpha and store the T_{af} results in an array. The next step was to input those results into the <u>**fit**</u> function of MATLAB and inputting a custom equation to find the relation between $T_{af} - T_r$ and α following equation (4). This piece of code was made into a function that was called <u>**curvefit**</u> that takes in a value of T_r and outputs the results of the curve fit along with the adiabatic temperature values for each alpha. The next step to this Task was to loop the <u>**curvefit**</u> function over three different reactant temperatures and plot the results. Finally, the accuracy of the curve fit was evaluated by using it to predict $T_{af} - T_r$ and then calculating the maximum and average deviation to the actual values of $T_{af} - T_r$.

Task 3

In this task we were asked to derive the relations for the mole fraction of each exhaust gas species as a function of molar air to fuel ratio α. The chemical reaction 5 we are dealing with is as shown below.

$$\begin{aligned}\left(\frac{\alpha}{4.76}\right)(O_2 + 3.76N_2) + [\gamma C_3H_8 + (1-\gamma)CH_4] \longrightarrow \\ \left(\frac{3.76\alpha}{4.76}\right)N_2 + \left(\frac{\alpha}{4.76} - 3\gamma - 2\right)O_2 + (2+2\gamma)H_2O + (1+2\gamma)CO_2\end{aligned} \tag{5}$$

The molar fractions for these product elements can be calculated by taking the ratio of the respective coefficients of each element to the sum of all the coefficients in the chemical reaction above. These numbers add up to 1. We then use these molar fractions to find out the $c_{p,prod}$ using the following equation 6.

$$\hat{c}_{p,prod} = y_{H_2O} * \hat{c}_{p,H_2O} + y_{CO_2} * \hat{c}_{p,CO_2} + y_{N_2} * \hat{c}_{p,N_2} + y_{O_2} * \hat{c}_{p,O_2} \tag{6}$$

We get the individual c_p values for each element from the function written in Homework 2. The calculated values for c_p for the products and air are summarized in the table below.

Table 3: Task3 - Results comparison at 600K

	cp-air	**cp-prod**
Values in kJ/kmol-K	30.46	35.63

Appendix shows the code for Task 3. The function **mol__fraction__finder** takes in the values of α and γ and outputs the mol fractions for all the product elements of combustion. Then the **cp__prod__finder** function takes in these mol fractions as well as the temperature and outputs the specific heat value for the products of combustion at that temperature.

Task 4

In this task, we are asked to perform a system analysis for specified $P_1 = 101kPa, T_1 = 298K, P_2 = 500kPa, T_4 = 1600K, \gamma = 0.25, \dot{Q}_S = 0, \eta_{comp} = 0.85, \eta_{turb} = 0.85$, and $\epsilon_{regen} = 0.75$.

Step 1: We first started by using the iterative scheme developed in Task 1 to determine the temperature and air molar enthalpy at the outlet of the compressor.

Step 2: Using the combustion equation below, we calculated the molar input rates of air and fuel ($\dot{n}_{air}$ and $\dot{n}_{fuel}$) and molar output rate of product ($\dot{n}_{prod}$). $\dot{n}_{air}$ was calculated by dividing the air mass flow rate by the molar mass of air. $\dot{n}_{fuel}$ and $\dot{n}_{prod}$ were calculated by taking appropriate ratios of molar fractions.

$$\begin{gathered}\left(\frac{\alpha}{4.76}\right)(O_2+3.76N_2)+[\gamma C_3H_8+(1-\gamma)CH_4]\longrightarrow \\ \left(\frac{3.76\alpha}{4.76}\right)N_2+\left(\frac{\alpha}{4.76}-3\gamma-2\right)O_2+(2+2\gamma)H_2O+(1+2\gamma)CO_2\end{gathered} \tag{7}$$

Step 3: We then used the function developed in Task 3 to calculate the molar fractions of the product gases using the value of $\alpha = \alpha_{stoic}$.
Step 4: We then guess a value of T_5 which is the temperature at the outlet of the turbine.
Step 5: Now, we compute the value of $\hat{c}_{p,prod}$ at the average temperature $T_{avg} = 0.5(T_4 + T_{5,guess})$. We then compute $\hat{c}_{v,prod} = \hat{c}_{p,prod} - \bar{R}$ and compute the change in enthalpy $\hat{h}_4 - \hat{h}_5$ using the following equation 8.

$$\hat{h}_4 - \hat{h}_5 = c_{p,prod}(T_4 - T_{5,guess}) \tag{8}$$

We then calculate T_5 using the following equation 9.

$$\frac{T_5}{T_4} = \frac{P_2}{P_1}^{\frac{k-1}{k}} \tag{9}$$

We then iterate until this calculated value of T_5 until it is not greater than the $T_{5,guess}$ value by 0.2.
Step 6: We then evaluate $\hat{c}_{p,air}$ and $\hat{c}_{p,prod}$ for $T_{avg,turb} = 0.5(T_2 + T_5)$. We then compute T_{2r} using the following equation 10.

$$\epsilon_{regen} = \frac{\dot{n}_{air}\hat{c}_{p,air}(T_{2r} - T_2)}{\dot{n}\hat{c}_{p,min}(T_5 - T_2)} \tag{10}$$

where $\dot{n}\hat{c}_{p,min}$ is the minimum value between $\dot{n}_{air}\hat{c}_{p,air}$ and $\dot{n}_{prod}\hat{c}_{p,prod}$. $\dot{n}_{prod}$ is calculated for an air mass flow rate of 6 kg/s.
Step 7: We determine T_3 using the following equation 11 for given solar input.

$$\dot{Q}_{solar} = \dot{n}_{air}\hat{c}_{p,air}(T_3 - T_{2r}) \tag{11}$$

Step 8: We then use the curve fit relation used in Task 2 to evaluate α given T_3 and T_4.
Step 9: We then use this new value of α to redo steps 2 to 8 until the value of α converges.
Step 10: We then compute the net power output, heat input into the combustor and the cycle efficiency using the value of α calculated above.

$$\dot{W}_{c,act} = \dot{n}_{prod}\hat{c}_{p,prod}(T_2 - T_1) \tag{12}$$

$$\dot{W}_{t,act} = \dot{n}_{prod}\hat{c}_{p,prod}(T_4 - T_5) \tag{13}$$

$$\dot{W}_{net} = \dot{W}_{t,act} - \dot{W}_{c,act} \tag{14}$$

$$\dot{Q}_c = \dot{n}_{prod}\hat{c}_{p,prod}(T_4 - T_3) \tag{15}$$

$$\eta_{cycle} = \dot{W}_{net}/\dot{Q}_c \tag{16}$$

The results are shown in Table 4.

Table 4: Cycle performance results

T_5 (K)	W_{net} (MW)	$\dot{Q}_s$ (MW)	η_{eff}	Constant c_p η_{eff}
1211	2.3	5.8	40 %	81.4 %

Appendix shows the flowchart explaining the code used for this Task. The script can be effectively explained by the steps above. It was then turned into a function that we called **findalpha**. The function took η_{comp}, η_{turb}, ϵ_{regen}, P_2, γ, T_4 and $\dot{Q}_s$. It returned α, T_5 and the performance metrics listed in Table 4.

Task 5

In this task, we were asked to analyze the system for different solar heat inputs to see how the efficiency of the system varies along with how the air to fuel ratio in the combustor changes. We did this by using the program developed for Task 4 to put in different values of solar heat input and observed how the cycle efficiency, net power output, and the heat input in the combustor varied. The results are shown below in Table 5 and Figures 6 and 7.

Table 5: α and cycle performance for different values of $\dot{Q}_s$

$\dot{Q}_s$	α	W_{net} (MW)	$\dot{Q}_s$ (MW)	η_{eff}
0	55.35	2.30	5.81	39.56
2	112.05	2.27	2.99	45.47
2.5	147.77	2.26	2.28	47.30
3	214.66	2.25	1.57	49.31

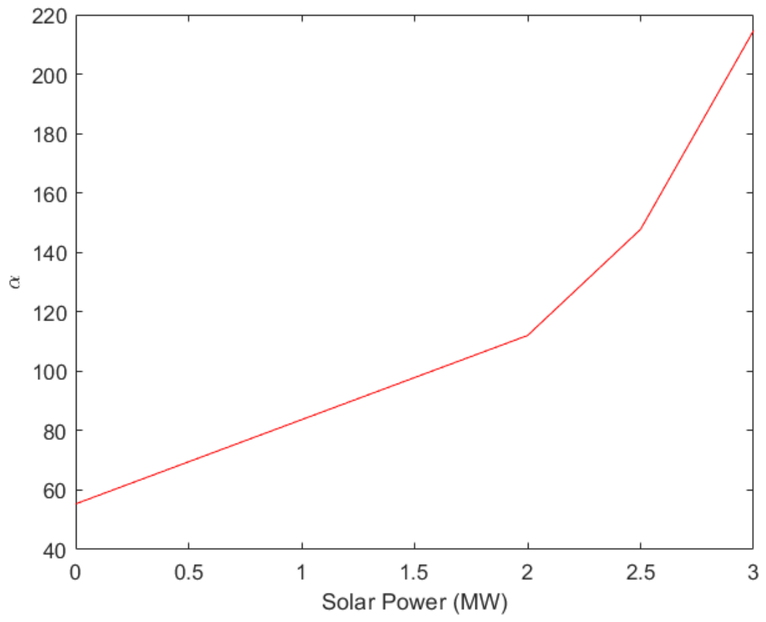

Figure 6: α as a function of $\dot{Q}_s$

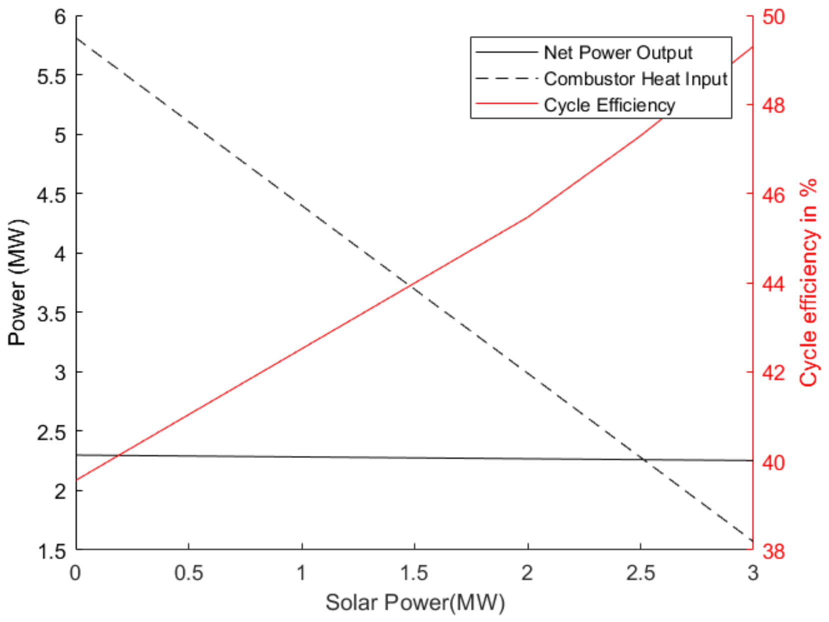

Figure 7: W_{net}, Q_c and η_{eff} as a function of $\dot{Q}_s$

For this task, the code simply took the **findalpha** function and looped it around the four values of $\dot{Q}_s$, stored the calues in arrays and plotted them.
We then analyzed the system for different propane to methane ratios from $\gamma = 0.5$ to 0. The results are shown in Figures 8, 9, 10, 11.

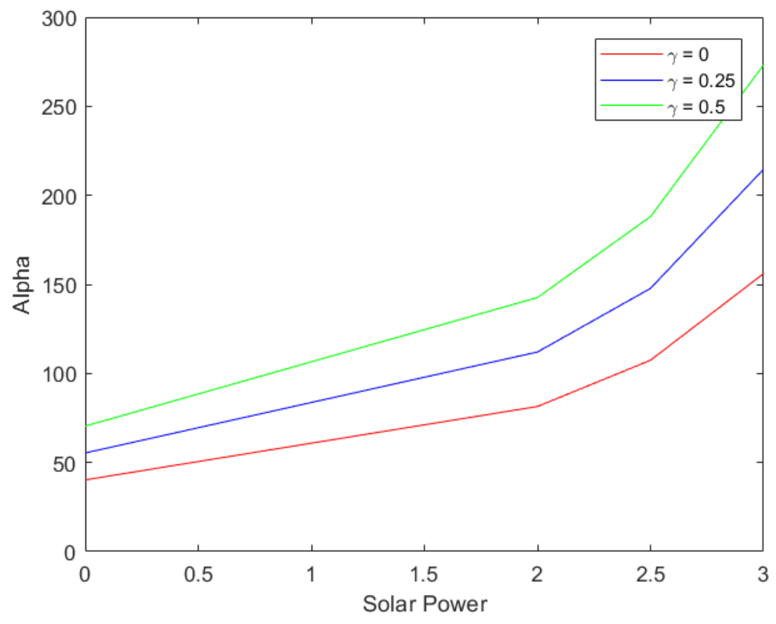

Figure 8: α as a function of $\dot{Q}_s$ for $\gamma = 0$, 0.25 and 0.5

As you can see, the value of α increases with the value of γ as it takes more air to combust the fuel as the amount of propane is increased in the mixture.

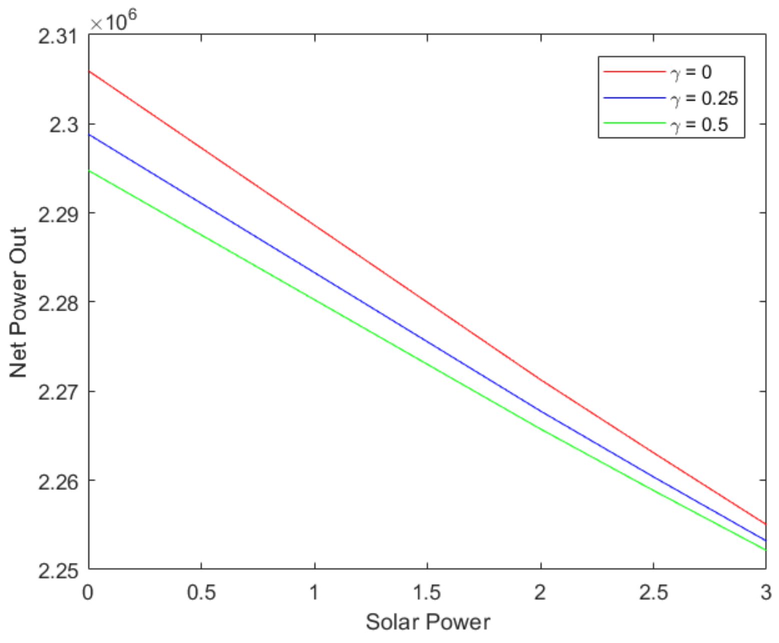

Figure 9: W_{net} as a function of $\dot{Q}_s$ for $\gamma = 0$, 0.25 and 0.5

Also, the net power output is the maximum when the mixture has no propane.

This is because, methane has a higher LHV value compared to propane as it has a higher ratio of H.

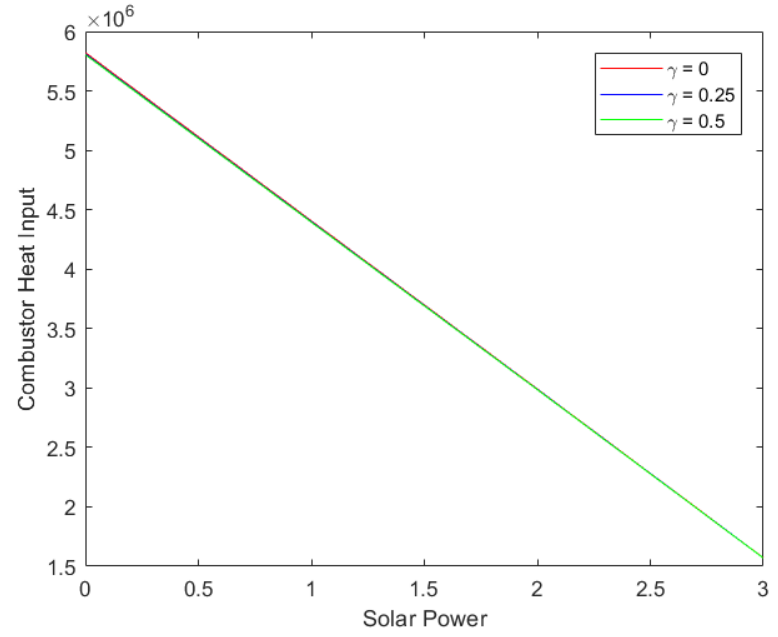

Figure 10: Q_c as a function of $\dot{Q}_s$ for $\gamma = 0$, 0.25 and 0.5

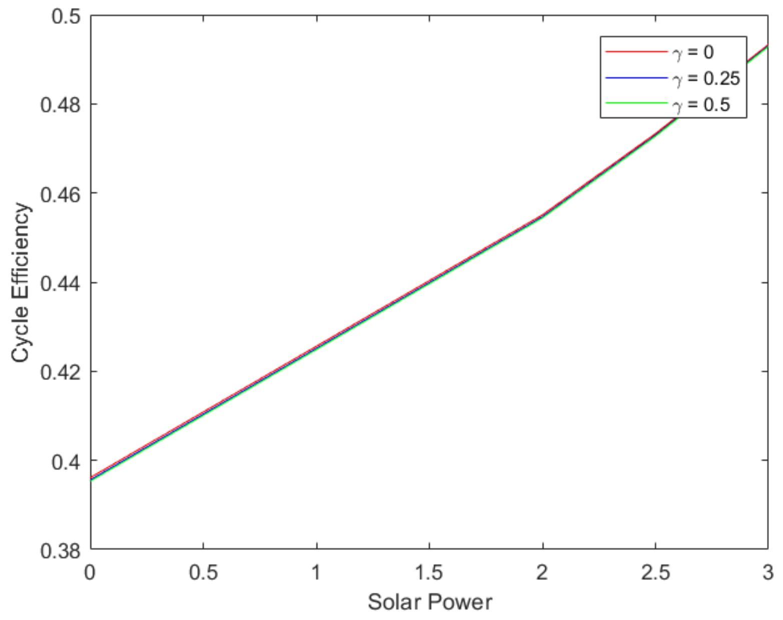

Figure 11: η_{eff} as a function of $\dot{Q}_s$ for $\gamma = 0$, 0.25 and 0.5

The cycle efficiency in this case remains the same as we are keeping the solar heat input constant. Therefore, the increase in γ results in consumption of more fuel but outputs the same deficit of heat after the solar heater to bring the turbine inlet temperature to 1600K.

For this task (part b), the code simply took the **findalpha** function and looped it in two nested loops: the first around three values of $\gamma = 0$, 0.25 and 0.5; the second around the four values of $\dot{Q}_s$.

The solar input is best utilized if the solar heater is dealing with air instead of the combustion products since the specific heat of air is always lesser than the specific heat of the products at the range of temperatures that we have analyzed in this project. Therefore, if we were to switch the solar input after the combustor, for the same amount of solar heat input, we would get lesser temperature increase as the solar heater will be dealing with the higher specific heat of the combustion products. Another problem with the placement of the solar input after the combustor would be the difficulty in controlling the input temperature into the turbine. In the original arrangement, the combustor can be easily throttled for different air to fuel ratios to make up for the variety in solar input throughout the day.

Appendix: Code and Flowcharts

Function: findConstants

```
function [a, b, c, d, h_i] = findConstants(i)

if i == 'o2--'
    a = 25.48;
    b = 1.520e-2;
    c = -0.7155e-5;
    d = 1.312e-9;
    h_i = 0;
elseif i == 'n2--'
    a = 28.9;
    b = -0.1571e-2;
    c = 0.8081e-5;
    d = -2.873e-9;
    h_i = 0;
elseif i == 'h2o-'
    a = 33.24;
    b = 0.1923e-2;
    c = 1.055e-5;
    d = -3.595e-9;
    h_i = -241820;
elseif i == 'c3h8'
    a = -4.04;
    b = 30.48e-2;
    c = -15.72e-5;
    d = 31.74e-9;
    h_i = -103850;
elseif i == 'ch4-'
    a = 19.89;
    b = 5.024e-2;
    c = 1.269e-5;
    d = -11.01e-9;
    h_i = -74850;
elseif i == 'co2-'
    a = 22.26;
    b = 5.981e-2;
    c = -3.501e-5;
    d = 7.469e-9;
    h_i = -393520;
elseif i == 'air-'
    a = 28.11;
    b = 0.1967e-2;
    c = 0.4802e-5;
    d = -1.966e-9;
end
```

Function: t2__W__h2__finder

```
function [t_2, W, h_2] = t2_W_h2_finder(p_2, eff_comp)
t_1 = 25 + 273;
t_2_guess = t_1 + 1; % left guess t2 in Kelvin
t_2_right_guess = 500 + 273; % right guess t2 in Kelvin
[a_air, b_air, c_air, d_air] = findConstants('air-');
p_1 = 101000; %inlet pressure in pascals
e = 2;
h_1 = 0;
while e > 0.2
    t_avg_comp = (t_1+t_2_guess)*0.5;
    cp_air = a_air + b_air*t_avg_comp + c_air*t_avg_comp^2 + d_air*t_avg_comp^3;
    R = 8.315; %gas constant in J/mol K
    c_v = cp_air - R;
    h_2 = cp_air*(t_2_guess - t_1) + h_1;
    k = cp_air / c_v;
    t_2 = t_1*(p_2/p_1)^((k-1)/k);
    e = abs(t_2_guess - t_2);
    t_2_guess = t_2;
end
t_2 = t_2_guess;
W = cp_air*(t_2 - t_1)/eff_comp;
```

Function: Task 2

```
clear all
close all

gamma = 0.25;
alpha_stoic = 4.76*(2+ 3*gamma);
i=1;
for T_r = [300,550,800]
[xData, yData, fitresult, a, n, T_af_array, alpha_array, T] = curvefit(T_r);
a_array(i) = a;
n_array(i) = n;
figure(1)
loglog(alpha_array, T)
hold on
xlabel('\alpha')
ylabel('T_{af} - T_r')
legend('300K', '500K', '800K')
T_pred = a.*(alpha_array./alpha_stoic).^(-n);
max_error = max(abs(T - T_pred));
mean_error = mean(abs(T - T_pred));
fprintf('The maximum absolute deviation for T_r = %f is %f \n', T_r, max_error)
fprintf('The mean absolute deviation for T_r = %f is %f \n', T_r, mean_error)
i = i+1;
end

i = 1;
for T_r = [300,550,800]
[xData, yData, fitresult, a, n, T_af_array, alpha_array] = curvefit(T_r);
T = T_af_array - T_r;
if i == 1
    dataformat = 'xb';
    curveformat = '-b';
elseif i == 2
    dataformat = 'xr';
    curveformat = '-r';
elseif i == 3
    dataformat = 'xg';
    curveformat = '-g';
end
figure(2)
plot(fitresult,curveformat, xData, yData,dataformat);
hold on
xlabel('\alpha')
ylabel('T_{af} - T_r')
legend('data 300K', 'curve fit 300K', 'data 550K', 'curve fit 550K', 'data 800K', 'curve fi
i = i + 1;
end
```

Function: curvefit

```
function [xData, yData, fitresult, a, n, T_af_array, alpha_array,T] = curvefit(T_r)
```

```
[a_o2, b_o2, c_o2, d_o2, h_i_o2] = findConstants('o2--');
[a_n2, b_n2, c_n2, d_n2, h_i_n2] = findConstants('n2--');
[a_h2o, b_h2o, c_h2o, d_h2o, h_i_h2o] = findConstants('h2o-');
[a_c3h8, b_c3h8, c_c3h8, d_c3h8, h_i_c3h8] = findConstants('c3h8');
[a_ch4, b_ch4, c_ch4, d_ch4, h_i_ch4] = findConstants('ch4-');
[a_co2, b_co2, c_co2, d_co2, h_i_co2] = findConstants('co2-');
alpha_array = [13.09, 16, 20, 24, 28, 32, 36];

i=1;
gamma = 0.25;
alpha_stoic = 4.76*(2+ 3*gamma);


for alpha = alpha_array
syms T_p h_o2_tr h_n2_tr h_c3h8_tr h_ch4_tr h_n2_tp h_o2_tp h_h2o_tp h_co2_tp

eqns = [h_o2_tr == h_i_o2 + (a_o2 + b_o2*(0.5*(T_r + T_p)) + c_o2*(0.5*(T_r + T_p))^2 + d_o

        h_n2_tr == h_i_n2 + (a_n2 + b_n2*(0.5*(T_r + T_p)) + c_n2*(0.5*(T_r + T_p))^2 + d_n

        h_c3h8_tr == h_i_c3h8 + (a_c3h8 + b_c3h8*(0.5*(T_r + T_p)) + c_c3h8*(0.5*(T_r + T_p

        h_ch4_tr == h_i_ch4 + (a_ch4 + b_ch4*(0.5*(T_r + T_p)) + c_ch4*(0.5*(T_r + T_p))^2

        h_n2_tp == h_i_n2 + (a_n2 + b_n2*(0.5*(T_r + T_p)) + c_n2*(0.5*(T_r + T_p))^2 + d_n

        h_o2_tp == h_i_o2 + (a_o2 + b_o2*(0.5*(T_r + T_p)) + c_o2*(0.5*(T_r + T_p))^2 + d_o

        h_h2o_tp == h_i_h2o + (a_h2o + b_h2o*(0.5*(T_r + T_p)) + c_h2o*(0.5*(T_r + T_p))^2

        h_co2_tp == h_i_co2 + (a_co2 + b_co2*(0.5*(T_r + T_p)) + c_co2*(0.5*(T_r + T_p))^2

        0 == (((alpha / 4.76) * h_o2_tr + ((3.76 * alpha) / 4.76) * h_n2_tr + gamma * h_c3h
        - ((((3.76 * alpha) / 4.76) * h_n2_tp + ((alpha / 4.76) - 3*gamma - 2) * h_o2_tp +

S = solve(eqns, [T_p h_o2_tr h_n2_tr h_c3h8_tr h_ch4_tr h_n2_tp h_o2_tp h_h2o_tp h_co2_tp]
T_af = double(S.T_p);
T_af_array(i) = T_af(1);
i = i+1;
end


T = T_af_array - T_r;

[xData, yData] = prepareCurveData( alpha_array./alpha_stoic, T);
ft = fittype( 'a*(x)^(-b)', 'independent', 'x', 'dependent', 'y' );
opts = fitoptions( 'Method', 'NonlinearLeastSquares' );
[fitresult, gof] = fit( xData, yData, ft, opts );
coeff = coeffvalues(fitresult);
a = coeff(1);
n = coeff(2);
```

Function: mol_frac_finder

```
function [y_H2O, y_CO2, y_N2, y_O2] = mol_frac_finder(alpha, gamma)

moles_H2O = 3.76 * alpha / 4.76;
moles_CO2 = ((alpha / 4.76) - (3 * gamma) - 2);
moles_N2 = 2 + 2 * gamma;
moles_O2 = 1 + 2 * gamma;
moles_total = moles_H2O + moles_CO2 + moles_N2 + moles_O2;
y_H2O = moles_H2O / moles_total;
y_CO2 = moles_CO2 / moles_total;
y_N2 = moles_N2 / moles_total;
y_O2 = moles_O2 / moles_total;
```

Function: cp_prod_finder

```
function [cp_prod] = cp_prod_finder(y_H2O, y_CO2, y_N2, y_O2, T_4)

[a_o2, b_o2, c_o2, d_o2] = findConstants('o2--');
[a_n2, b_n2, c_n2, d_n2] = findConstants('n2--');
[a_h2o, b_h2o, c_h2o, d_h2o] = findConstants('h2o-');
[a_co2, b_co2, c_co2, d_co2] = findConstants('co2-');

cp_O2 = (a_o2 + b_o2*(T_4) + c_o2*(T_4)^2 + d_o2*(T_4)^3);
cp_H2O = (a_h2o + b_h2o*(T_4) + c_h2o*(T_4)^2 + d_h2o*(T_4)^3);
cp_N2 = (a_n2 + b_n2*(T_4) + c_n2*(T_4)^2 + d_n2*(T_4)^3);
cp_CO2 = (a_co2 + b_co2*(T_4) + c_co2*(T_4)^2 + d_co2*(T_4)^3);

cp_prod = cp_O2 * y_O2 + cp_CO2 * y_CO2 + cp_N2 * y_N2 + cp_H2O * y_H2O;
```

Function: Task 3

```
clear all

gamma = 0.25;
alpha_stoic = 4.76*(2+ 3*gamma);
alpha = alpha_stoic;
T = 600;

[y_H2O, y_CO2, y_N2, y_O2] = mol_frac_finder(alpha, gamma);
[cp_prod] = cp_prod_finder(y_H2O, y_CO2, y_N2, y_O2, T);

fprintf("The molar fraction of the products are as follows:\n");
fprintf("\nCO2: %1.4f",y_CO2);
fprintf("\nH2O: %1.4f",y_H2O);
fprintf("\nO2: %1.4f",y_O2);
fprintf("\nN2: %1.4f",y_N2);

fprintf("\ncp_prod: %4.4f",cp_prod);
```

Function: findalpha

```
function [alpha, t_5, w_t_actual, w_c_actual, w_net, q_c, eff_cycle, eff_cycle_constant_cp

i = 1;
e2 = 1;
alpha_stoic = 4.76*(2+ 3*gamma);
alpha_guess = alpha_stoic;
while e2 > 0.2


[t_2, W, h_2] = t2_W_h2_finder (p_2, eff_comp);
t_1 = 25 + 273;
R = 8.315; %gas constant in J/mol K
p_1 = 101000; %inlet pressure in pascals

e = 1;
t_5_guess = 1000;


m_dot_air = 6; %in kg/s
molar_mass_air = 0.02897; %kg/mol
n_dot_air = (m_dot_air/molar_mass_air); %mol/s
n_dot_fuel = n_dot_air / alpha_guess;
moles_prod_per_fuel = alpha_guess + gamma + 1; % moles of product per moles of fuel
n_dot_prod = n_dot_fuel * moles_prod_per_fuel; %mol/s

    while e>0.2
        [y_H2O, y_CO2, y_N2, y_O2] = mol_frac_finder(alpha_guess, gamma);
        t_avg_turbine = 0.5*(t_4 + t_5_guess);
        [cp_prod] = cp_prod_finder(y_H2O, y_CO2, y_N2, y_O2, t_avg_turbine); %kJ/kmol K
        cv_prod = cp_prod - R;
        k = cp_prod / cv_prod;
        delta_h5_h4 = cp_prod*(t_4 - t_5_guess);
        p_4 = p_2;
        p_5 = p_1;
        t_5 = t_4*(p_5/p_4)^((k-1)/k);
        e = abs(t_5 - t_5_guess);
        t_5_guess = t_5;
    end


[a_air, b_air, c_air, d_air] = findConstants('air-');
cp_air = a_air + b_air*t_avg_turbine + c_air*t_avg_turbine^2 + d_air*t_avg_turbine^3; %kJ/

ncp_air = n_dot_air*cp_air; %J/sK or W/K
```

```
ncp_prod = n_dot_prod*cp_prod; %J/sK or W/K

ncp_min = min(ncp_air,ncp_prod); %J/sK or W/K

t_2r = ((eff_reg * ncp_min * (t_5_guess - t_2)) / ncp_air) + t_2;

t_3 = (q_dot_sol / (ncp_air)) + t_2r;

[xData, yData, fitresult, a, n, T_af_array, alpha_array] = curvefit(t_3);
alpha_new = ((t_4 - t_3)/a)^(-1/n) * alpha_stoic;
e2 = abs(alpha_new - alpha_guess);
alpha_guess = alpha_new;
i = i+1;
end
alpha = alpha_guess;
cv_air = cp_air - R;
k_air = cp_air/cv_air;
k_prod = cp_prod/cp_prod;

w_t_actual = cp_prod*(t_4-t_5)*n_dot_prod*eff_turbine;
w_c_actual = cp_air*(t_2-t_1)*n_dot_air*eff_comp;
w_net = w_t_actual - w_c_actual;
q_c = cp_prod*(t_4 - t_3)*n_dot_prod;
eff_cycle = w_net / (q_c + q_dot_sol);
eff_cycle_constant_cp = 1 - (t_1/t_4)*(p_2/p_1)^((k_prod-1)/k_prod);
```

Function: Task 4

```
clear all
close all
clc

eff_comp = 0.85;
eff_turbine = 0.85;
eff_reg = 0.75;
p_2 = 500000;
gamma = 0.25;
t_4 = 1600;
q_dot_sol = 0;
[alpha, t_5, w_t_actual, w_c_actual, w_net, q_c, eff_cycle,...
    eff_cycle_constant_cp] = findalpha(eff_comp, eff_turbine,...
    eff_reg, p_2, gamma, t_4, q_dot_sol);
```

Function: Task 5

```
clear all
close all
```

```
clc

eff_comp = 0.85;
eff_turbine = 0.85;
eff_reg = 0.75;
p_2 = 500000;
gamma = 0.25;
t_4 = 1600;

alpha_array = [];
w_net_array = [];
q_c_array = [];
eff_cycle = [];

i = 1;
for q_dot_sol = [0, 2e6, 2.5e6, 3e6]

    [alpha, t_5, w_t_actual, w_c_actual, w_net, q_c, eff_cycle,...
        eff_cycle_constant_cp] = findalpha(eff_comp, eff_turbine,...
        eff_reg, p_2, gamma, t_4, q_dot_sol);

    alpha_array(i) = alpha;
    w_net_array(i) = w_net;
    q_c_array(i) = q_c;
    eff_cycle_array(i) = eff_cycle;

    i = i+1;
end

figure(1)
plot([0,2,2.5,3], alpha_array, '-r')
xlabel('Solar Power(MW)')
ylabel('alpha')

fig = figure(2);
left_color = [0 0 0];
right_color = [1 0 0];
set(fig,'defaultAxesColorOrder',[left_color; right_color]);
hold on
yyaxis left
plot([0,2,2.5,3], w_net_array.*1e-6, '-k')
plot([0,2,2.5,3], q_c_array.*1e-6, '--k')
ylabel('Power (MW)')
xlabel('Solar Power(MW)')
yyaxis right
plot([0,2,2.5,3], eff_cycle_array.*100, '-r')
ylabel('Cycle efficiency in %')
legend('Net Power Output','Combustor Heat Input','Cycle Efficiency')
```

Function: Task 5b

```
clear all
close all
clc

eff_comp = 0.85;
eff_turbine = 0.85;
eff_reg = 0.75;
p_2 = 500000;
t_4 = 1600;

alpha_array = [];
w_net_array = [];
q_c_array = [];
eff_cycle = [];


i = 1;
for gamma = [0,0.25,0.5]
for q_dot_sol = [0, 2e6, 2.5e6, 3e6]

    [alpha, t_5, w_t_actual, w_c_actual, w_net, q_c, eff_cycle, eff_cycle_constant_cp] = f

    alpha_array(i) = alpha;
    w_net_array(i) = w_net;
    q_c_array(i) = q_c;
    eff_cycle_array(i) = eff_cycle;

    i = i+1;
end
end

figure(1)
plot([0,2,2.5,3], alpha_array(1:4), '-r',...
[0,2,2.5,3], alpha_array(5:8), '-b',...
[0,2,2.5,3], alpha_array(9:12), '-g')
xlabel('Solar Power')
ylabel('Alpha')
legend('\gamma = 0', '\gamma = 0.25', '\gamma = 0.5')
figure(2)
plot([0,2,2.5,3], w_net_array(1:4), '-r',...
[0,2,2.5,3], w_net_array(5:8), '-b',...
[0,2,2.5,3], w_net_array(9:12), '-g')
xlabel('Solar Power')
ylabel('Net Power Out')
legend('\gamma = 0', '\gamma = 0.25', '\gamma = 0.5')
figure(3)
plot([0,2,2.5,3], eff_cycle_array(1:4), '-r',...
[0,2,2.5,3], eff_cycle_array(5:8), '-b',...
[0,2,2.5,3], eff_cycle_array(9:12), '-g')
xlabel('Solar Power')
ylabel('Cycle Efficiency')
legend('\gamma = 0', '\gamma = 0.25', '\gamma = 0.5')
figure(4)
plot([0,2,2.5,3], q_c_array(1:4), '-r',...
[0,2,2.5,3], q_c_array(5:8), '-b',...
[0,2,2.5,3], q_c_array(9:12), '-g')
xlabel('Solar Power')
ylabel('Combustor Heat Input')
legend('\gamma = 0', '\gamma = 0.25', '\gamma = 0.5')
```

Task 1 Flowchart

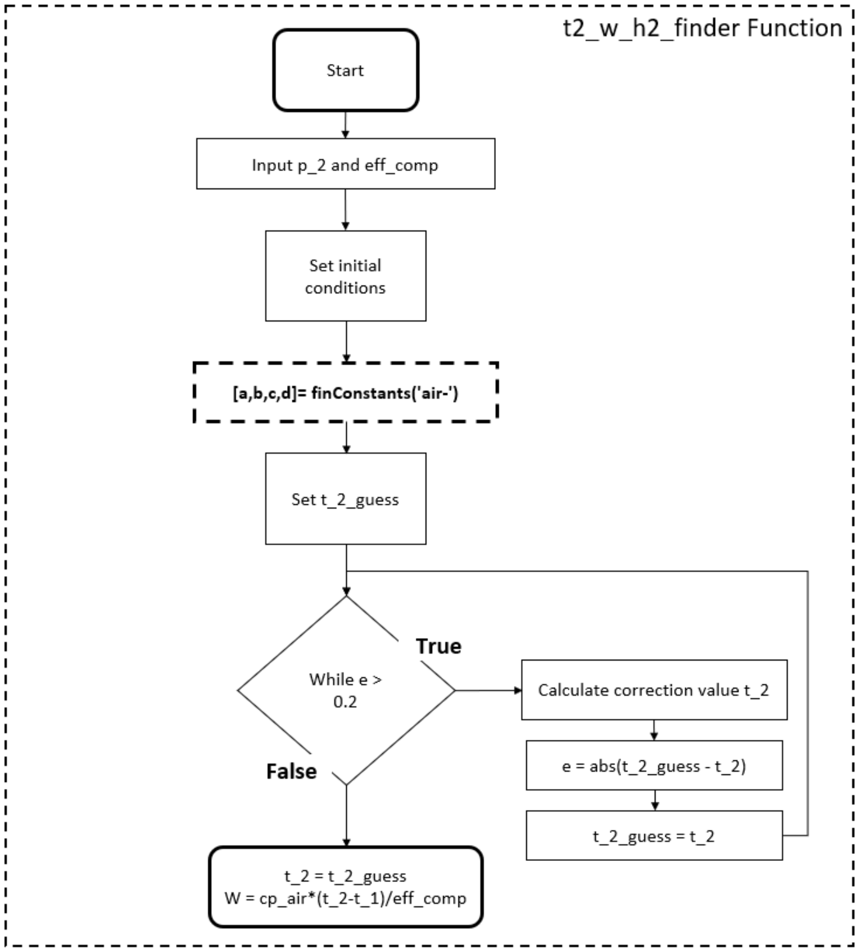

Figure 12: **t2__W__h2__finder** Flowchart

Task 2 Flowchart

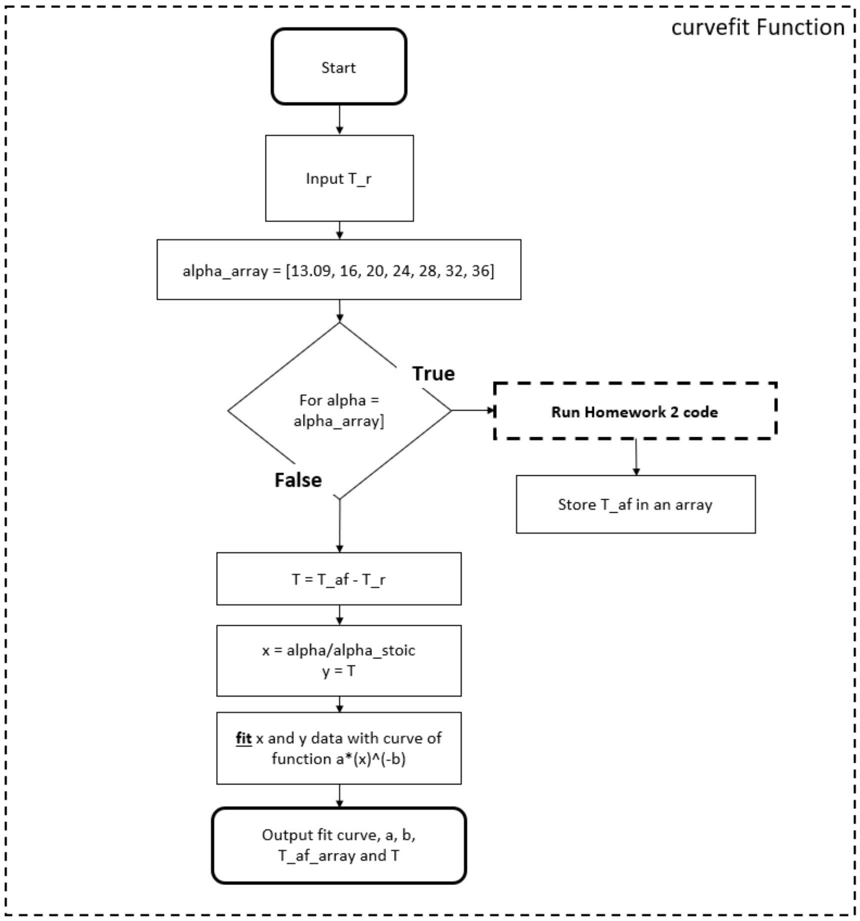

Figure 13: **curvefit** Flowchart

Task 4 and 5 Flowchart

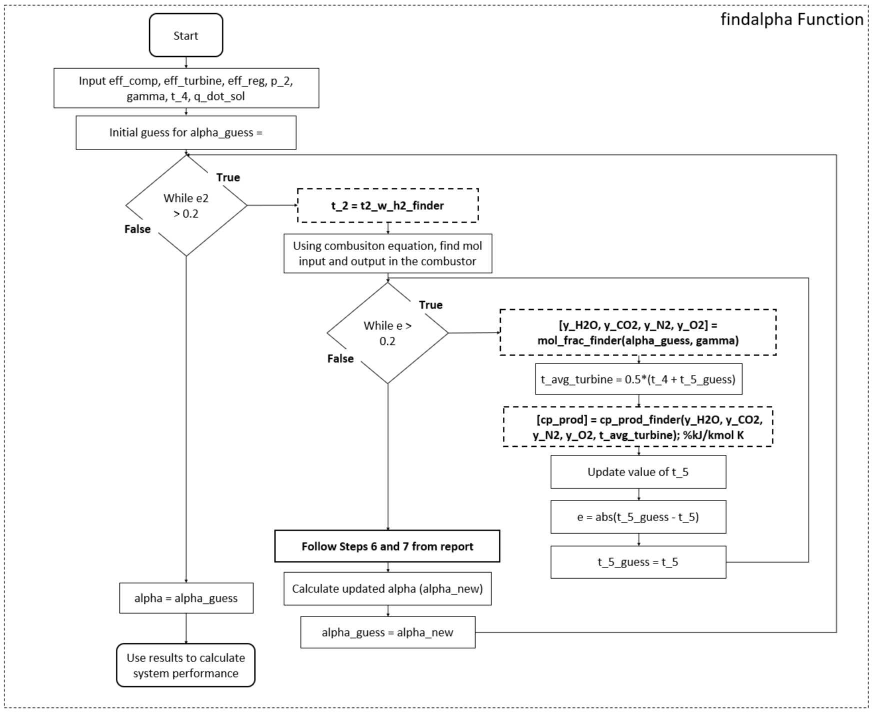
findalpha Function
Start
Input eff_comp, eff_turbine, eff_reg, p_2, gamma, t_4, q_dot_sol
Initial guess for alpha_guess =
While e2 > 0.2
True
False
t_2 = t2_w_h2_finder
Using combusiton equation, find mol input and output in the combustor
While e > 0.2
True
False
[y_H2O, y_CO2, y_N2, y_O2] = mol_frac_finder(alpha_guess, gamma)
t_avg_turbine = 0.5*(t_4 + t_5_guess)
[cp_prod] = cp_prod_finder(y_H2O, y_CO2, y_N2, y_O2, t_avg_turbine); %kJ/kmol K
Update value of t_5
e = abs(t_5_guess - t_5)
t_5_guess = t_5
Follow Steps 6 and 7 from report
Calculate updated alpha (alpha_new)
alpha_guess = alpha_new
alpha = alpha_guess
Use results to calculate system performance

Figure 14: **findalpha** Flowchart