MAE 563 Project: Propulsion System Analysis Tool

Anushka Subedi

ASU ID: 1225812200

Contents

1 Introduction

This project aims to analyze ramjet and scramjet engines using a MATLAB-based parametric tool. The tool models non-ideal flow conditions for engines with a fixed-geometry converging-only nozzle, enabling systematic variation of input parameters such as flight altitude, Mach number, diffuser and nozzle efficiencies, and combustor temperature limits. The propulsion system is as shown in 1 with all the states 1-2-3-e-4.

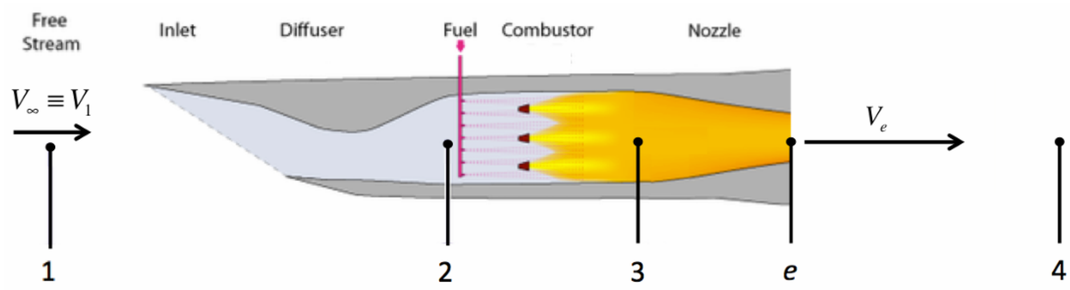

Figure 1: Propulsion System

The following inputs are used in the analysis tool:

Input Parameter	Symbol	Units
Flight altitude	z	meters
Flight Mach number	M_1	–
Inlet/diffuser efficiency	η_d	–
Mach number at diffuser exit	M_2	–
Maximum allowable total temperature at combustor exit	$(T_{t3})_{max}$	K
Fuel heating value	q_f	J/kg
Nozzle efficiency	η_n	–
Nozzle exit area	A_e	m^2

These inputs are used to calculate key outputs at each engine state, such as temperature, pressure, Mach number, and velocity. The outputs Calculated at each state 1-2-3-e-4:

Output Parameter	Symbol
Total and static pressure and temperature	P_t, T_t, P, T
Relative entropy change	$\Delta s = s_i - s_{i-1}$
Flow speed	V
Mach number	M

After getting the outputs at each step, they are used to find the thrust, power and efficiencies of the propulsion system. The outputs calculated at the end of the Propulsion System:

Output Parameter	Symbol	Units
Thrust	T	N
Propulsive power	P	W
Thermal efficiency	η_{th}	–
Propulsive efficiency	η_p	–
Overall efficiency	η_o	–
Exit mass flow rate	$\dot{m}_e$	kg/s
Fuel mass flow rate	$\dot{m}_f$	kg/s
Thrust specific fuel consumption	$TSFC$	(kg/hr)/N
Specific impulse	I_{sp}	s

The modules in the analysis tool:

Module Number	Description
Module 1	Isentropic Atmosphere
Module 2	Inlet/Diffuser
Module 3	Combustor
Module 4	Nozzle
Module 5	External Nozzle
Module 6	Outputs

Two validation cases: Non-thermally choked case and Thermally choked case were used to validate the developed modules before going into the analysis parts A-I.

2 Part A: T-S diagrams

The T-s diagrams for first validating case (non-thermally choked) and second validating (thermally choked) are presented below. These diagrams are crucial for evaluating the performance of Brayton Cycle engines, as the area enclosed by the T-s curve represents the net work produced by the cycle. In Figure2, corresponding to the non-thermally choked case, the significantly larger enclosed area highlights that thermal choking limits performance by capping the maximum achievable combustor temperature.

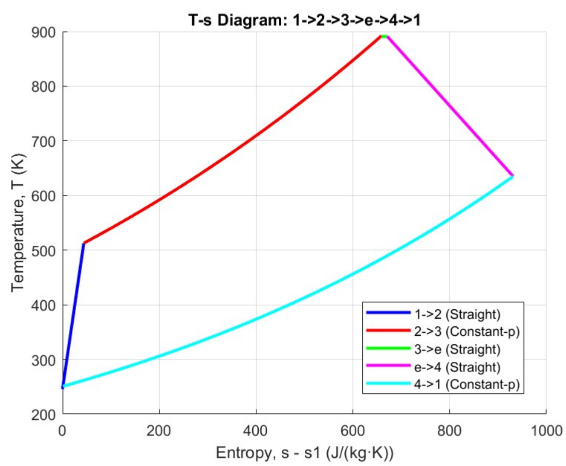

Figure 2: T-S diagram for thermally un-choked case

The gaps in the T-s diagrams arise due to the method used to calculate the entropy changes in the project. Specifically, the integrated form of the second law of thermodynamics was applied, but the specific heat capacity (Cp) was not properly accounted for as a function of temperature (T). Instead of using the mean value theorem (MVT) value of Cp, which adjusts for the variation of Cp with temperature, the project instructions used Cp(T)=a+bT corresponding to a single temperature (T3). This simplification fails to fully capture the actual entropy changes across each step. At this point of the project, the T-S diagrams are presented as they are. However, the correct approach which involves integrating Cp(T) over the temperature range to calculate entropy changes and the MVT value of Cp can be used in future, which ensure that the temperature dependence is properly accounted for, eliminating discrepancies.

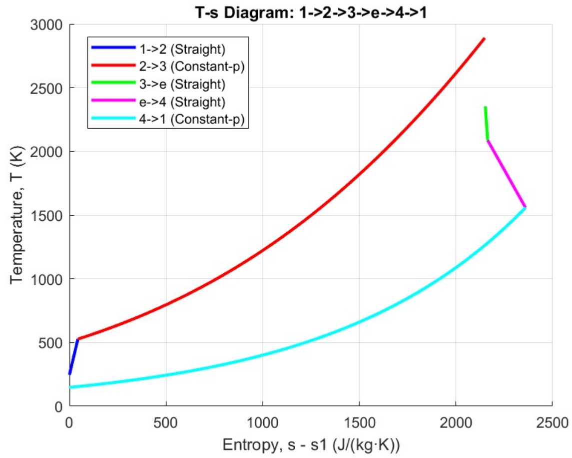

Figure 3: T-S diagram for thermally choked case

3 Part B: Standard and Isentropic Atmosphere

The flow path analysis starts by determining atmospheric conditions at the engine's operating flight envelope. This analysis uses the isentropic atmosphere model to calculate temperatures and pressures at various altitudes, as represented by Equations (1)-(4). A comparison with International Standard Atmosphere (ISA) data shows that pressure values are nearly identical, while the isentropic model slightly underestimates temperature at most altitudes. Despite these differences, the isentropic model is reliable and suitable for this analysis, as it aligns well with regional temperature variations for the altitudes of interest.

For flight altitudes $z < 7958\,\mathrm{m}$:

$$\frac{T(z)}{T_s} = \left[1 - \frac{\gamma - 1}{\gamma}\left(\frac{z}{z^*}\right)\right], \tag{1}$$

$$\frac{P(z)}{P_s} = \left[1 - \frac{\gamma - 1}{\gamma}\left(\frac{z}{z^*}\right)\right]^{\frac{\gamma}{\gamma-1}}. \tag{2}$$

For flight altitudes $z > 7958\,\mathrm{m}$:

$$T(z) = 210\,\mathrm{K} \tag{3}$$

$$P(z) = 33.6\,\mathrm{e}^{-\frac{z-7958}{6605}}. \tag{4}$$

$$T_s = 288\,\mathrm{K}, \quad P_s = 101.3 \times 10^3\,\mathrm{Pa}, \quad \gamma = 1.4, \quad z^* = 8404\,\mathrm{m}.$$

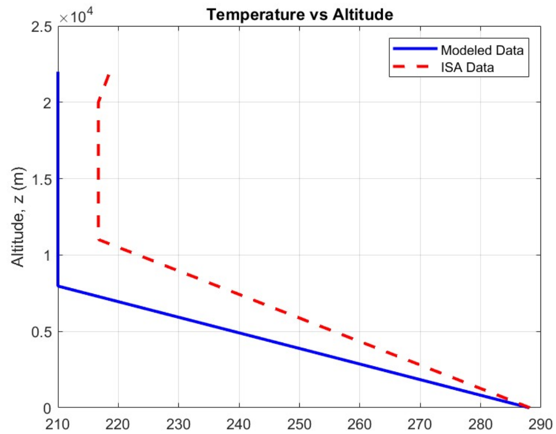

Figure 4: Temperature VS altitude

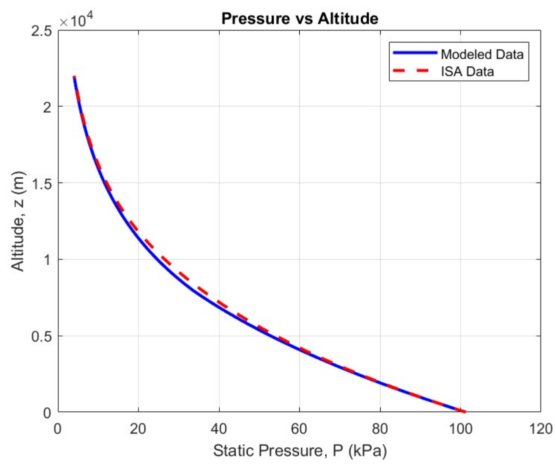

Figure 5: Pressure VS altitude

4 Part C: Study of M1 Variation

In part C, the inputs from validation case (a) are utilized, with the flight Mach number, M1 varied incrementally from 0.8 to 5 to examine its impact on overall efficiency, thrust, and thrust-specific fuel consumption.

4.1 Overall efficiency

The overall efficiency with Mach number M1 is as shown in Figure 6. The overall efficiency is low to begin with because the diffuser struggles to generate sufficient ”ram compression” at lower flight speeds. It slowly goes up with it being the highest when the flight Mach number is between 3 and 3.5. Again, the overall efficiency drops sharply at flight speeds exceeding Mach 4. This behavior is influenced by the condition that the combustor inlet Mach number is fixed at 0.15 (non-thermally choked), requiring air traveling at supersonic speeds to decelerate significantly to Mach 0.15 at the combustor inlet.

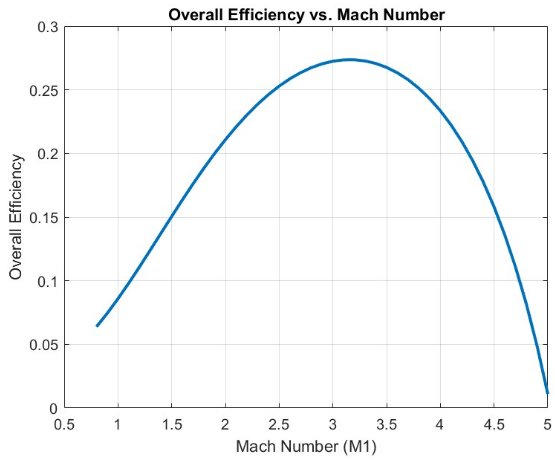

Figure 6: Overall efficiency as a function of M1

4.2 Thrust

The thrust with Mach number M1 is as shown in Figure 7. The thrust increases with increasing M1, peaking at Mach 4 to 4.5. After Mach 4.5, it drops sharply.

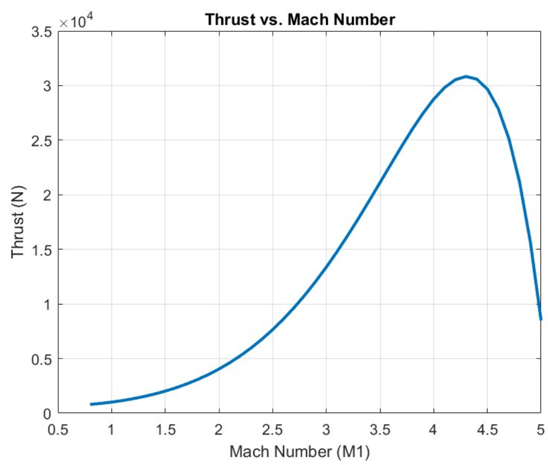

Figure 7: Thrust as a function of M1

4.3 TSFC

The TSFC with Mach number M1 is as shown in Figure 8. We can see that the TSFC remains mostly stable up to a flight speed of about Mach 4, after which it rises sharply. This behavior makes sense because flying at very high supersonic speeds requires a significant increase in fuel flow to maintain performance. This trend is also reflected in the overall efficiency plot, highlighting the challenges of operating at such high velocities. There is no clear "best" flight speed to operate a ramjet engine, as every speed involves trade-offs. The choice of flight speed should depend on the specific requirements of the mission, balancing efficiency, thrust, and fuel consumption to meet operational goals effectively.

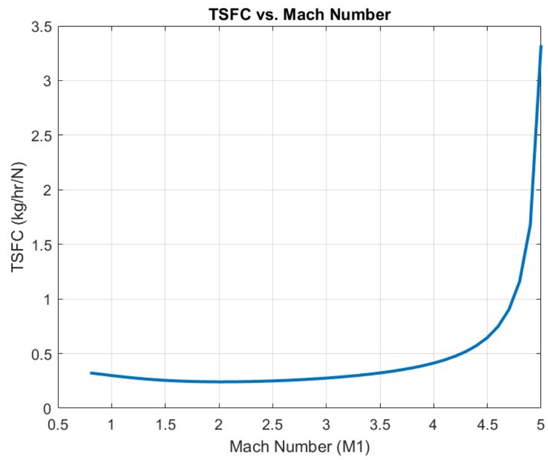

Figure 8: TSFC as a function of M1

5 Part D: Study of altitude Variation

In part D, the impact of flight altitude on performance parameters is analyzed by gradually increasing altitude, z from 2,000 to 30,000 meters.

5.1 Overall efficiency

The overall efficiency with altitude z is as shown in Figure 9. The overall efficiency decreases almost linearly up to an altitude of 8,000 meters, after which it remains constant up to 30,000 meters. This trend can be explained by temperature plot in standard vs isentropic atmosphere study, which illustrates that beyond 8,000 meters, the temperature stays constant in the isentropic atmosphere.

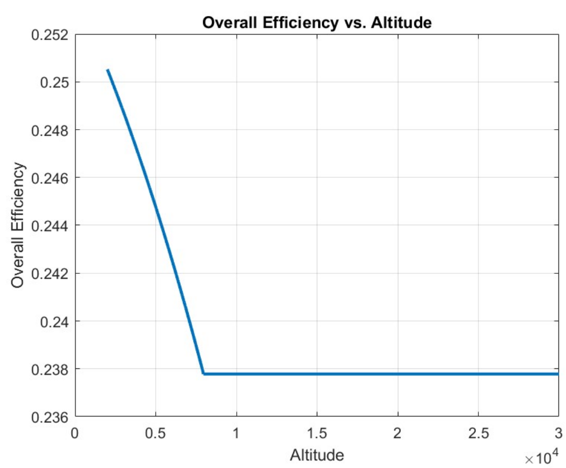

Figure 9: Overall efficiency vs altitude

5.2 Thrust

The thrust with altitude z is as shown in Figure 10.The thrust decreases with increasing altitude. This happens because higher altitudes have lower air density, reducing the mass flow rate into the engine.

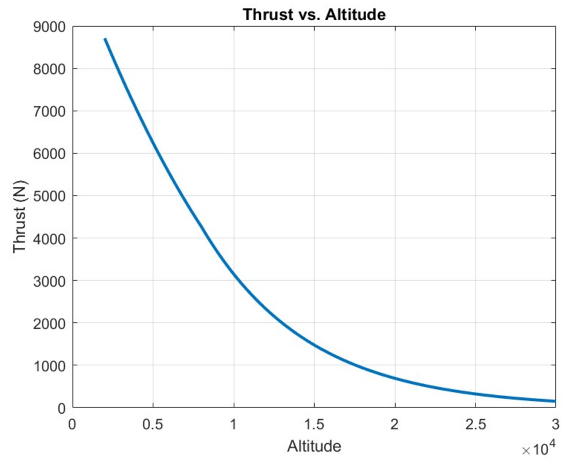

Figure 10: Thrust vs altitude

5.3 TSFC

The TSFC with altitude z is as shown in Figure 11. The TSFC has the same pattern as that for the overall efficiency and for the same reasons.

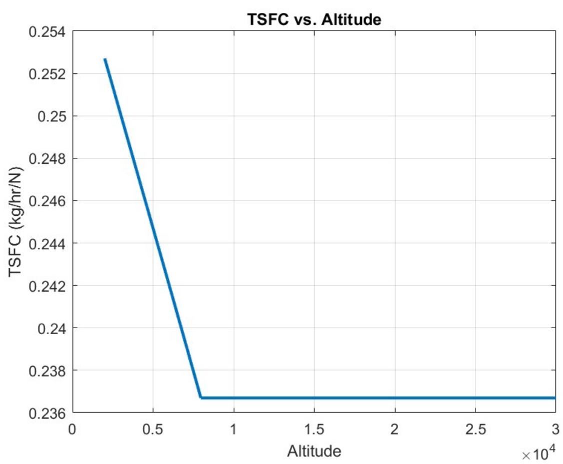

Figure 11: TSFC vs altitude

6 Part E: Study of M1 and altitude Variation

In part E, the flight altitude was varied from 2,000 to 20,000 meters in 500-meter increments. For each altitude, the flight Mach number was adjusted to find the value that maximized overall efficiency and this optimal Mach number was recorded. A similar approach was taken to identify the flight Mach number that minimized Thrust Specific Fuel Consumption (TSFC).

6.1 Optimized Overall efficiency

The overall efficiency optimizing (maximum) Mach number for each altitude are as shown in Figure 12. From 2,000 to 8,000 meters, the optimal flight Mach number increases linearly with altitude. Beyond 8,000 meters, it levels off, stabilizing at approximately 3.35 for the remainder of the altitude range.

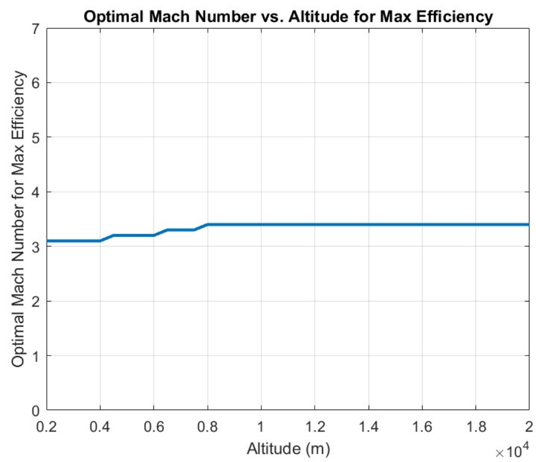

Figure 12: M1 VS Altitude for optimized thrust

6.2 Optimized TSFC

The TSFC optimizing (minimum) Mach number for each altitude are as shown in Figure 13. The trend for TSFC is different than that for overall efficiency, showing a sharp drop in the optimal flight Mach number between 2,000 and 4,000 meters. At lower altitudes, the denser atmosphere allows for higher thrust due to increased air mass flow through the engine. However, as altitude increases, achieving higher thrust becomes more fuel-intensive, causing the optimal flight Mach number to stabilize around 2 for the rest of the range.

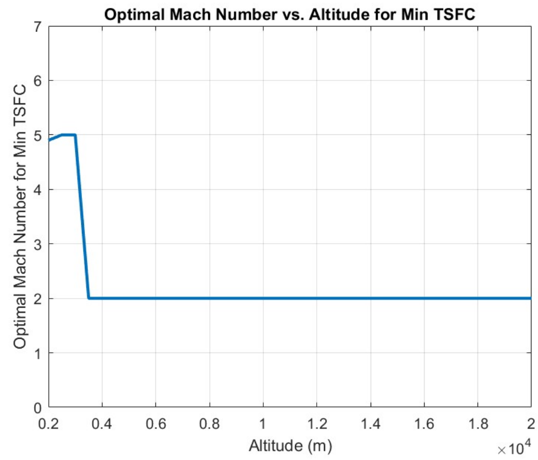

Figure 13: M1 VS Altitude for optimized thrust

7 Part F: Study of diffuser efficiency Variation

Part F highlights the significant impact of irreversibility by incrementally increasing diffuser efficiency and observing its effect on various performance parameters. The diffuser efficiency is varied from 0.5 to 1, representing the isentropic condition.

7.1 Overall efficiency

The overall efficiency with diffuser efficiency is as shown in Figure 14. The overall efficiency improves as ram compression becomes more efficient. This is because with higher diffuser efficiency, total pressure loss through the diffuser decreases, contributing to better overall performance.

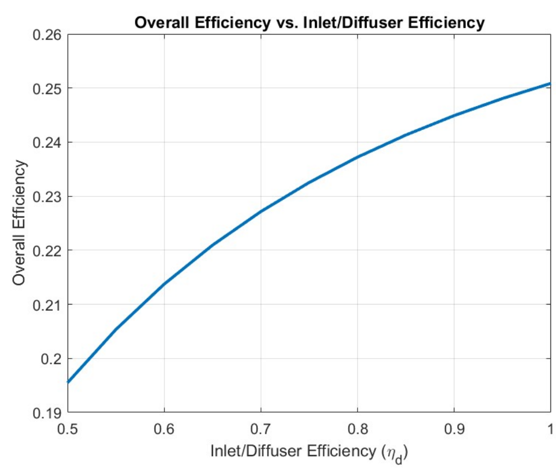

Figure 14: Overall efficiency vs diffuser efficiency

7.2 Thrust

The thrust with diffuser efficiency is as shown in Figure 15. The thrust increase greatly with increasing diffuser efficiency. The thrust produced with an ideal diffuser is nearly four times greater than that of a diffuser with 0.5 efficiency. This is because the ideal diffuser retains a much higher pressure ratio, significantly enhancing performance.

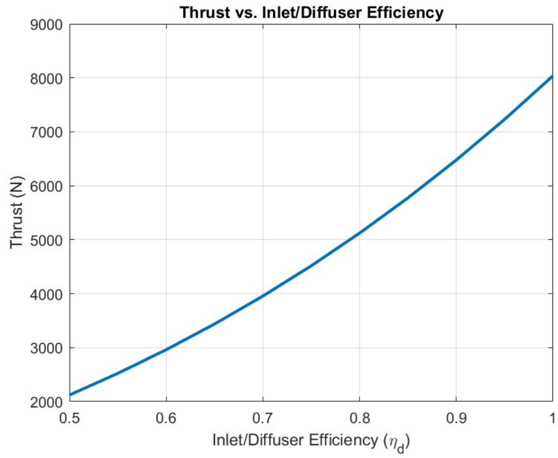

Figure 15: Thrust vs diffuser efficiency

7.3 TSFC

The TSFC with diffuser efficiency is as shown in Figure 16. TSFC decreases significantly as the diffuser becomes more efficient, approaching ideal performance. This is because TSFC is inversely proportional to thrust and has opposite effects to that for thrust.

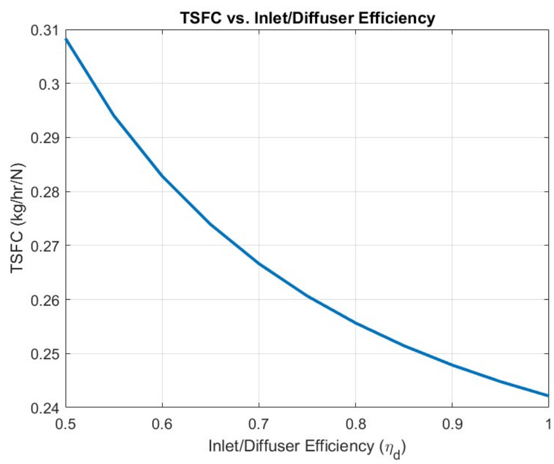

Figure 16: TSFC vs diffuser efficiency

8 Part G: Study of nozzle efficiency Variation

The part G examines the effect of varying the efficiency of the converging nozzle from 0.5 to 1 (isentropic).

8.1 Overall efficiency

The overall efficiency with nozzle efficiency is as shown in Figure 17. The overall efficiency improves as compression becomes more efficient. This is because with higher nozzle efficiency, total pressure loss through the nozzle decreases, contributing to better overall performance.

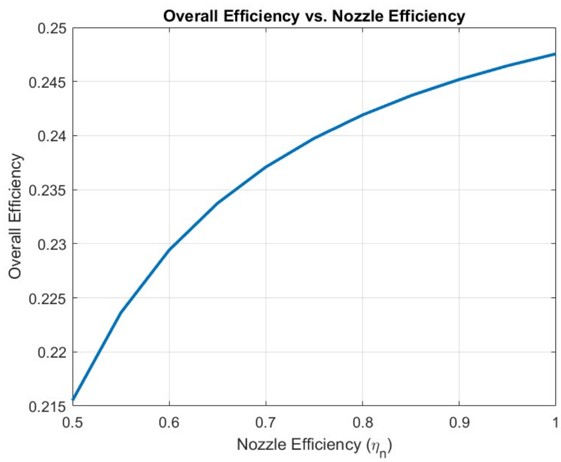

Figure 17: Overall efficiency vs nozzle efficiency

8.2 Thrust

The thrust with nozzle efficiency is as shown in Figure 18. The thrust increases with increasing nozzle efficiency. The thrust produced with an ideal nozzle is nearly 2.5 times greater than that of a nozzle with 0.5 efficiency. This is because the ideal nozzle retains a much higher pressure ratio, enhancing performance but is lower than the results from diffuser efficiency.

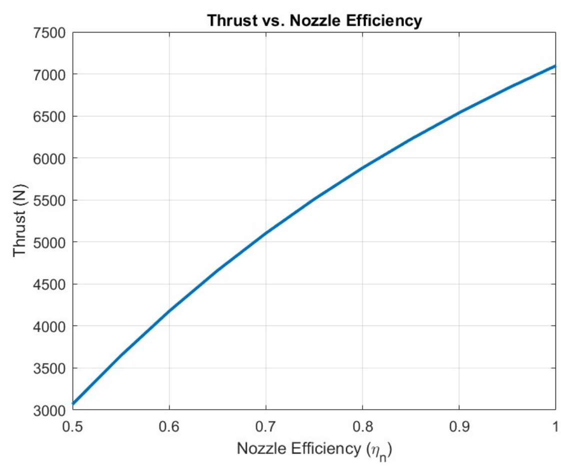

Figure 18: Thrust vs nozzle efficiency

8.3 TSFC

The TSFC with nozzle efficiency is as shown in Figure 19. TSFC decreases as the nozzle becomes more efficient, approaching ideal performance. This is because TSFC is inversely proportional to thrust and has opposite effects to that for thrust

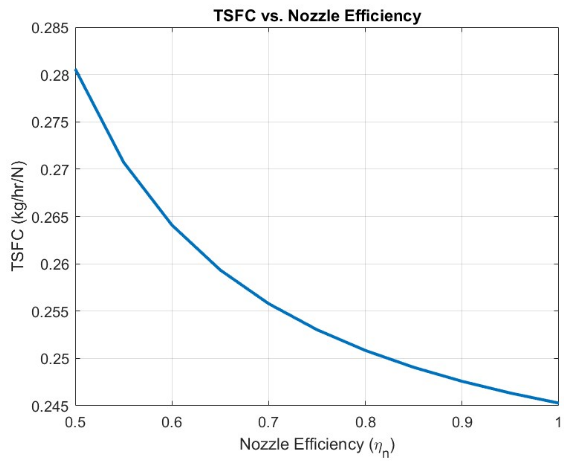

Figure 19: TSFC vs nozzle efficiency

9 Part H: Study of M2 Variation

In part H, the combustor inlet Mach number, M2 is varied from 0.1 to 2.5, and its impact on overall efficiency, thrust, and thrust-specific fuel consumption is analyzed.

9.1 Overall efficiency

The overall efficiency with M2 is as shown in Figure 20. The combustor inlet Mach number beyond about 0.6 until 1.3 results in a negative overall efficiency which is not useful.

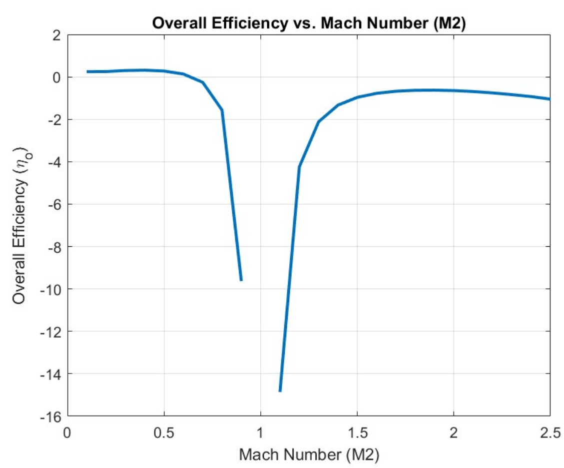

Figure 20: Overall efficiency vs M2

9.2 Thrust

The thrust with M2 is as shown in Figure 21. Thrust performance declines sharply when the combustor inlet Mach number exceeds approximately 0.3. The unfavorable behavior of performance parameters closely tied to the combustor becoming thermally choked around M2=0.3, as evident from the validation case 2 with M2=0.4. At this point, additional heat input no longer raises combustion temperatures, limiting the performance.

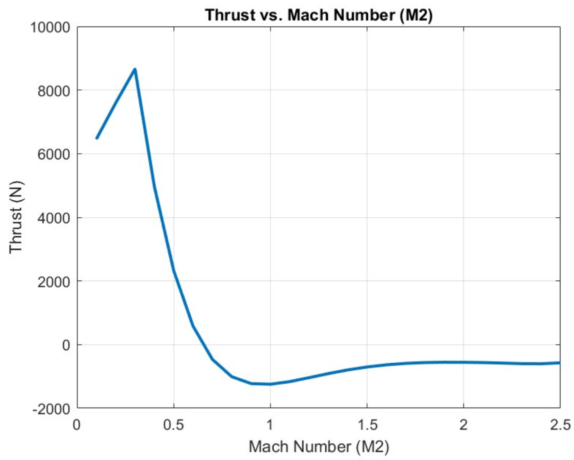

Figure 21: Thrust vs M2

9.3 TSFC

The TSFC with M2 is as shown in Figure 22. Thrust Specific Fuel Consumption (TSFC) displays unrealistic and erratic values when the combustor inlet Mach number, M2 exceeds approximately 0.6. This behavior indicates a deviation from efficient operation at higher Mach numbers. But TSFC does show an optimal range that closely aligns with the thrust performance (valley in the thrust peak) trends observed in Figure 21.

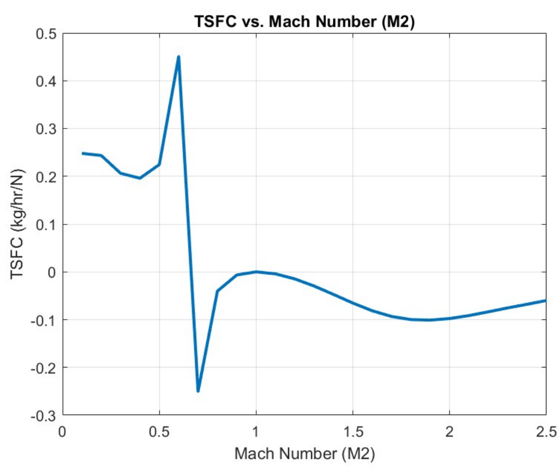

Figure 22: TSFC vs M2

10 Part I: Design of Ramjet/Scramjet

Using the parametric analysis tool, a ram/scramjet propulsion system was designed for hypersonic flight at M1=5 and z= 90,000ft (27,400m), adhering to the specified constraints. The design maintained the same fuel heating value (qf) and inlet/diffuser and nozzle efficiencies as in the validation cases. The combustor exit total temperature (Tt3) was constrained to a maximum of 2400K, ensuring the diffuser exit Mach number (M2) met this requirement. By systematically varying M2 and Tt3, two optimal designs were identified: one that maximized positive thrust and another that maximized overall efficiency. The results for maximized positive thrust are presented in Table(1) and the results for maximized overall efficiency are listed in Table(2).

Maximum Thrust	M2	Tt3
1970.42 N	0.41	2360K

Table 1: Optimized Thrust

Maximum Overall Efficiency	M2	Tt3
0.1305	0.40	2400K

Table 2: Optimized Overall Efficiency

From the results, we can note that the design parameters for maximizing thrust and overall efficiency—specifically the combustor exit temperature and combustor inlet Mach number—are nearly identical. This outcome is logical, as the combustor exit temperature is capped at 2400 K, reflecting the realistic material limitations of modern engines. This temperature constraint significantly influences the extent to which the engine's performance and efficiency can be optimized.

11 Conclusion

This project provided a detailed analysis of ramjet and scramjet propulsion systems using a MATLAB-based parametric tool. By modeling non-ideal flow conditions and systematically varying key input parameters, the study explored the impact on performance metrics such as thrust, overall efficiency, and thrust-specific fuel consumption.

The analysis showed that increasing flight Mach number improves overall efficiency and thrust up to a peak between Mach 3 and 3.5. However, beyond Mach 4, efficiency drops due to limitations in the diffuser and combustor. Similarly, altitude variations revealed that overall efficiency decreases linearly up to 8,000 meters before stabilizing, reflecting the isentropic atmosphere's behavior at higher altitudes.

Improving diffuser and nozzle efficiencies significantly enhanced thrust and overall efficiency by reducing pressure losses. This also lowered thrust-specific fuel consumption (TSFC), highlighting the importance of minimizing irreversibility in propulsion systems to achieve better performance.

Varying the combustor inlet Mach number revealed important trade-offs. While performance improved within an optimal range, significant limitations occurred beyond M2=0.3 due to thermal choking, which restricted the combustor's ability to use additional heat effectively.

Finally, a ram/scramjet propulsion system was designed for hypersonic flight at M1=5 and z=27,400m. The optimal designs for maximum thrust and maximum overall efficiency showed that combustor exit temperature constraints play a key role in determining performance. Future advances in materials (by the whitecoat lab people :)) capable of handling higher temperatures could further enhance the performance of ramjet and scramjet engines, enabling more efficient designs for hypersonic applications.

Appendix

Code for all modules development:

```
clc;clear;clear all;
%% module 1

% Constants
gamma = 1.4;          % Specific heat ratio
R = 286.9;            % Specific gas constant in J/(kg*K)
z_star = 8404;        % Scale height in meters
T_s = 288.0;          % Standard temperature at sea level in Kelvin
p_s = 101.3;          % Standard pressure at sea level in kPa

% Input Parameters
z = 4300;
M1=2.4;

% Calculate T and P for each altitude
    if z < 7958  % Within the troposphere
        T1 = T_s * (1 - (((gamma - 1) / gamma) * (z / z_star)));
        p1 = p_s * ((1 - (((gamma - 1) / gamma) * (z / z_star)))^(gamma / (
            gamma - 1)));
    else              % In the tropopause
        T1 = 210.0;  % Constant temperature in tropopause
        p1 = 33.6 * exp(-(z - 7958) / 6605);
    end

% Total-to-static relations
Tt1 = T1 * (1 + (gamma - 1) / 2 * M1^2);
pt1 = p1 * (1 + (gamma - 1) / 2 * M1^2)^(gamma / (gamma - 1));

% Sound speed and velocity
a1 = sqrt(gamma * R * T1);
V1 = M1 * a1;
%% module 2

% Constants
gamma = 1.4;          % Specific heat ratio
R = 286.9;            % Specific gas constant in J/(kg*K)

% Inputs from Module 1 or given data
%M2 = 0.15;            % Mach number at State 2
M2=0.40;               %vary for case 2
eta_d = 0.92;         % Inlet/diffuser efficiency

% Module 2 Calculations
% Total temperature remains constant (no work or heat transfer)
Tt2 = Tt1;
```

```matlab

% Compute static temperature at State 2
T2 = Tt2 / (1 + (gamma - 1) / 2 * M2^2);

% Compute total and static pressures
pt2 = p1 * (1 + (eta_d *(gamma - 1) / 2) * M1^2)^(gamma / (gamma - 1));
p2 = pt2 / (1 + (gamma - 1) / 2 * M2^2)^(gamma / (gamma - 1));

% Compute entropy change across the diffuser
cp2 = 1004; % Specific heat at constant pressure (J/(kg*K)) for air
Delta_s_12 = cp2* log(Tt2 / Tt1) - R * log(pt2 / pt1); % Entropy change (J/(kg*K))

% Compute speed of sound and velocity at State 2
a2 = sqrt(gamma * R * T2); % Speed of sound at State 2
V2 = M2 * a2;              % Velocity at State 2

%% module 3

% Constants
gamma = 1.3;                  % Specific heat ratio changed to 1.3 from 1.4
Tt3_max = 2400;                  % Maximum allowable total temperature at combustor exit (K)


% Module 3 Calculations
% Step 1: Check if the combustor is thermally choked
Tt3_choked = Tt2 * (1 / (2 * (gamma + 1))) * ...
                    ((1 / (M2^2) * ((1 + gamma* M2^2)^2))) * ...
                    ((1 + (((gamma - 1) / 2) * M2^2)))^(-1);

% If thermally choked, adjust the maximum temperature
if Tt3_choked < Tt3_max
    Tt3 = Tt3_choked;
    M3=1;
else
    Tt3 = Tt3_max;
    % Solve for M3 in the non-choked case
     % Use quadratic equation to solve for M3
     C = (Tt3 / Tt2) * ((1 + (gamma - 1) / 2 * M2^2) / ((1 + gamma * M2^2)^2)) * M2^2;


    % Quadratic coefficients
    a = C * (gamma^2) - ((gamma - 1)/2);
    b = 2 * C * gamma - 1;
    c = C;

   % Solve the quadratic equation
    M3_roots = roots([a, b, c]);

        if M2<1
        M3_squared = M3_roots(M3_roots > 0 & M3_roots <= 1);
        else
```

```
        M3_squared = M3_roots( M3_roots >=1);
        end


    % Take the square root to find M3
    M3 = sqrt(M3_squared);
end

% Step 2: Compute heat added (q23)
q23 = 986 * (Tt3 - Tt2)+0.5*0.179*(Tt3^2 - Tt2^2);

% Step 3: Compute static properties at combustor exit
T3 = Tt3 / (1 + (gamma - 1) / 2 * M3.^2);
p3 = p2;  % Static pressure remains the same in constant-pressure combustion
pt3 = p3 * ((1 + (gamma - 1) / 2 * M3.^2).^(gamma/(gamma-1)));


% Step 4: Compute speed of sound and velocity at combustor exit
a3 = sqrt(gamma * R * T3); % Speed of sound at State 3
V3 = M3 * a3;                              % Velocity at State 3

%need cp3 from T3
cp3=986+0.179*T3;

% Step 5: Compute entropy increase across the combustor
Delta_s_23 = cp3* log(Tt3 / Tt2) - R * log(pt3 / pt2);

%entropy chnage 1-3
Delta_s_31 = cp3* log(Tt3 / Tt1) - R * log(pt3 / pt1);

%% module 4

% Inputs
Ae=0.015;
eta_n=0.94;
Tte = Tt3;                    % Total temperature at nozzle entrance


% Step 1: Compute test Mach number
test_M = sqrt(2 / (gamma - 1)) * sqrt((eta_n * ...
          ((1 - (p1 / pt3)^((gamma - 1) / gamma))) / ...
          (1 - eta_n * (1- (p1 / pt3)^((gamma - 1) / gamma)))));


% Step 2: Check choking condition
if test_M < 1
    % Nozzle is not choked
    Me = test_M;
    pe = p1;  % Exit pressure equals ambient pressure
else
    % Nozzle is choked
    Me = 1;
    pe = pt3 * (1 - (1 / eta_n) * (gamma - 1) / (gamma + 1))^(gamma / (gamma -
        1));
```

```
end

% Step 3: Compute static properties at nozzle exit
Te = Tte / (1 + (gamma - 1) / 2 * Me^2); % Static temperature
pte= pe*((1 + (gamma - 1) / 2 * Me^2)^(gamma / (gamma - 1)));

%Compute speed of sound and velocity at nozzle exit
ae = sqrt(gamma * R * Te);                 % Speed of sound
Ve = Me * ae;                              % Velocity

% Step 4: Compute entropy increase
cpe = 986 + 0.179 * Te; % Specific heat at exit (J/(kg*K))
Delta_s_3e = cpe * log(Tte / Tt3) - R * log(pte / pt3);


% Mass flux
rho_e = pe *1000/ (R * Te); % Convert Pe to Pascals if needed
m_dot_e = rho_e * Ve * Ae;

%% module 5
Tt4=Tte;

% Step 2: Apply exit criterion for eta_n_ext
if test_M < 1
    eta_n_ext = 1;
else
    eta_n_ext = test_M.^ (-0.3);
end

% Step 3: Compute T4 (Static Temperature at State 4)
T4 = Tte * (1 - eta_n_ext * (1 - (p1 / pte)^((gamma - 1) / gamma)));

% Step 4: Compute M4 (Mach Number at State 4)
M4 = sqrt((2 / (gamma - 1)) * ((Tt4 / T4) - 1));


% Step 6: Compute p4 (Static Pressure at State 4)
p4 = p1; % Assume static pressure matches ambient pressure

% Step 7: Compute pt4 (Total Pressure at State 4)
pt4 = p4 * (1 + (gamma - 1) / 2 * M4^2)^(gamma / (gamma - 1));

% Step 8: Compute velocity at State 4
a4 = sqrt(gamma * R * T4);     % Speed of sound at State 4
V4 = M4 * a4;                 % Velocity at State 4

% Step 9: Compute entropy increase across the nozzle
cp4 = 986 + 0.179 * T4;
Delta_s_4e = cp4 * log(Tt4 / Tte) - R * log(pt4 / pte);
Delta_s_41 = Delta_s_12+Delta_s_23+Delta_s_3e+Delta_s_4e;

%% module 6

% Constants
```

```
g0 = 9.81;                    % Gravitational acceleration (m/s^2)
q_f = 43.2e6;                 % Heating value of fuel (J/kg)


% Step 1: Compute air mass flow rate (mi)
m_dot_i = m_dot_e / (1 + q23 / q_f);

% Step 2: Compute fuel mass flow rate (mf)
m_dot_f = m_dot_e - m_dot_i;

% Step 3: Compute fuel-to-air ratio (f)
f = m_dot_f / m_dot_i;

% Step 4: Compute thrust
jet_thrust = m_dot_i * (1 + f) * Ve - m_dot_i * V1; % Jet thrust
pressure_thrust = (pe - p1)*1000 * Ae;                  % Pressure thrust
total_thrust = jet_thrust + pressure_thrust;              % Total thrust

% Step 5: Compute equivalent velocity (Veq)
Veq = Ve + ((pe - p1) * 1000*Ae / m_dot_e);          % Equivalent velocity

% Step 6: Compute TSFC
TSFC = (m_dot_f / total_thrust) * 3600;                  % TSFC in (kg/hr)/N

% Step 7: Compute specific impulse
Isp = total_thrust / (m_dot_f * g0);                     % Specific impulse
    (s)

% Step 8: Compute efficiencies
thermal_efficiency = (m_dot_e * Veq^2 / 2 - m_dot_i * V1^2 / 2) / (m_dot_i *
    q23);
propulsive_efficiency = 2 / (1+(Veq /V1));
overall_efficiency = thermal_efficiency * propulsive_efficiency;

%Propulsive power
Prop_power=total_thrust*V1;
```

Code for part A:

```
clc; clear; close all;

% Constants
a = 986;                  % Coefficient for cp(T) fit
b = 0.179;                    % Constant coefficient for cp(T) fit
cp_4to1 = 1004;            % Constant cp for state 4->1 (J/kg-K)

% Static temperature (K) and entropy (J/(kg K)) at each state
%T = [245.9, 512.8, 891, 891, 635];
%s = [0, 43.95, 658.95, 670.95, 930.95];

T = [245.9, 526.8, 2354, 2087, 1558];
s = [0, 43.95, 2152, 2164, 2360]; %

% Extract points for states
T1 = T(1); T2 = T(2); T3 = T(3); Te = T(4); T4 = T(5);
```

```
s1 = s(1); s2 = s(2); s3 = s(3); se = s(4); s4 = s(5);

% Step sizes for entropy
ds_23 = 5; % Smaller step size for 2->3
ds_41 = 5; % Smaller step size for 4->1

% 1 -> 2 (straight line)
T_12 = linspace(T1, T2, 100);
s_12 = linspace(s1, s2, 100);

% Initialize entropy and temperature arrays
s_23 = s2:ds_23:s3; % Generate entropy steps
T_23 = zeros(size(s_23)); % Preallocate temperature array
T_23(1) = T2; % Set initial temperature

% Iteratively compute T values
for i = 2:length(s_23)
    % Compute change in entropy
    ds_actual = s_23(i) - s_23(i-1);

    % Update temperature using the relation
    dT = (T_23(i-1) * ds_actual) / (a + b * T_23(i-1));
    T_23(i) = T_23(i-1) + dT;
end



% 3 -> e (straight line)
T_3e = linspace(T3, Te, 100);
s_3e = linspace(s3, se, 100);

% e -> 4 (straight line)
T_e4 = linspace(Te, T4, 100);
s_e4 = linspace(se, s4, 100);

% 4 -> 1 (constant-pressure curve)
T_41 = T4; % Initialize with T4
s_41 = s4:-ds_41:s1; % Entropy steps
for i = 2:length(s_41)
    T_41(i) = T_41(i-1) - (T_41(i-1) * ds_41) / cp_4to1;
end

% Plot T-s diagram
figure;
hold on;

% Add lines and curves
plot(s_12, T_12, 'b', 'LineWidth', 2, 'DisplayName', '1->2 (Straight)');
plot(s_23, T_23, 'r', 'LineWidth', 2, 'DisplayName', '2->3 (Constant-p)');
plot(s_3e, T_3e, 'g', 'LineWidth', 2, 'DisplayName', '3->e (Straight)');
plot(s_e4, T_e4, 'm', 'LineWidth', 2, 'DisplayName', 'e->4 (Straight)');
plot(s_41, T_41, 'c', 'LineWidth', 2, 'DisplayName', '4->1 (Constant-p)');
```

```
% Set axes and labels
xlabel('Entropy, s - s1 (J/(kg K))');
ylabel('Temperature, T (K)');
title('T-s Diagram: 1->2->3->e->4->1');
legend('show');
grid on;
% hold off;
```

Code for part B:

```
clc;clear;clear all;
%% module 1

% Constants
gamma = 1.4;          % Specific heat ratio
R = 286.9;            % Specific gas constant in J/(kg*K)
z_star = 8404;        % Scale height in meters
T_s = 288.0;          % Standard temperature at sea level in Kelvin
p_s = 101.3;          % Standard pressure at sea level in kPa

% Input Parameters
z = linspace(0, 22000, 500);  % Altitude range from 0 to 25,000 meters

% Initialize arrays for temperature and pressure
T1 = zeros(size(z));
p1 = zeros(size(z));

% Calculate T and P for each altitude
for i = 1:length(z)
    if z(i) < 7958  % Within the troposphere
        T1(i) = T_s * (1 - (((gamma - 1) / gamma) * (z(i) / z_star)));
        p1(i) = p_s * ((1 - (((gamma - 1) / gamma) * (z(i) / z_star)))^(gamma / (gamma - 1)));
    else              % In the tropopause
        T1(i) = 210.0;  % Constant temperature in tropopause
        p1(i) = 33.6 * exp(-(z(i) - 7958) / 6605);
    end
end

% International Standard Atmosphere (ISA) data
z_isa = [0, 500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500, 5000, ...
         5500, 6000, 6500, 7000, 7500, 8000, 8500, 9000, 9500, 10000, ...
         10500, 11000, 11500, 12000, 12500, 13000, 13500, 14000, ...
         14500, 15000, 15500, 16000, 16500, 17000, 17500, 18000, ...
         18500, 19000, 19500, 20000, 22000];
T_isa = [288.15, 284.9, 281.7, 278.4, 275.2, 271.9, 268.7, 265.4, ...
         262.2, 258.9, 255.7, 252.4, 249.2, 245.9, 242.7, 239.5, ...
         236.2, 233, 229.7, 226.5, 223.3, 220, 216.8, 216.7, ...
         216.7, 216.7, 216.7, 216.7, 216.7, 216.7, 216.7, 216.7, ...
         216.7, 216.7, 216.7, 216.7, 216.7, 216.7, 216.7, 216.7, 216.7 218.6];
p_isa = [101.325, 95.46, 89.88, 84.56, 79.5, 74.69, 70.12, 65.78, ...
         61.66, 57.75, 54.05, 50.54, 47.22, 44.08, 41.11, 38.3, ...
         35.65, 33.15, 30.8, 28.58, 26.5, 24.54, 22.7, 20.98, ...
         19.4, 17.93, 16.58, 15.33, 14.17, 13.1, 12.11, 11.2, ...
         10.35, 9.572, 8.85, 8.182, 7.565, 6.995, 6.467, 5.98, 5.529, 4.047];
```

```

% Plot T vs z
figure;
plot(T1, z, 'b-', 'LineWidth', 2); % Modeled data
hold on;
plot(T_isa, z_isa, 'r--', 'LineWidth', 2); % ISA data
ylabel('Altitude, z (m)');
zlabel('Static Temperature, T (K)');
title('Temperature vs Altitude');
legend('Modeled Data', 'ISA Data');
grid on;

% Plot P vs z
figure;
plot(p1,z, 'b-', 'LineWidth', 2); % Modeled data
hold on;
plot(p_isa, z_isa, 'r--', 'LineWidth', 2); % ISA data
ylabel('Altitude, z (m)');
xlabel('Static Pressure, P (kPa)');
title('Pressure vs Altitude');
legend('Modeled Data', 'ISA Data');
grid on;
```

Code for part C:

```
clc; clear; clear all;

% Initialize Mach number range
M1_range = 0.8:0.1:5.0;

% Preallocate arrays for results
overall_efficiency_results = zeros(size(M1_range));
thrust_results = zeros(size(M1_range));
TSFC_results = zeros(size(M1_range));

for idx = 1:length(M1_range)
    % Set M1 for this iteration
    M1 = M1_range(idx);

    %% module 1
    % Constants
    gamma = 1.4; % Specific heat ratio
    R = 286.9; % Specific gas constant in J/(kg*K)
    z_star = 8404; % Scale height in meters
    T_s = 288.0; % Standard temperature at sea level in Kelvin
    p_s = 101.3; % Standard pressure at sea level in kPa

    % Input Parameters
    z = 4300;

    % Calculate T and P for each altitude
    if z < 7958 % Within the troposphere
        T1 = T_s * (1 - (((gamma - 1) / gamma) * (z / z_star)));
        p1 = p_s * ((1 - (((gamma - 1) / gamma) * (z / z_star)))^(gamma / (
            gamma - 1)));
```

```matlab
    else % In the tropopause
        T1 = 210.0; % Constant temperature in tropopause
        p1 = 33.6 * exp(-(z - 7958) / 6605);
    end

    % Total-to-static relations
    Tt1 = T1 * (1 + (gamma - 1) / 2 * M1^2);
    pt1 = p1 * (1 + (gamma - 1) / 2 * M1^2)^(gamma / (gamma - 1));

    % Sound speed and velocity
    a1 = sqrt(gamma * R * T1);
    V1 = M1 * a1;

    %% module 2
    gamma = 1.4; % Specific heat ratio
    eta_d = 0.92; % Inlet/diffuser efficiency
    M2 = 0.15;

    Tt2 = Tt1;
    T2 = Tt2 / (1 + (gamma - 1) / 2 * M2^2);
    pt2 = p1 * (1 + (eta_d * (gamma - 1) / 2) * M1^2)^(gamma / (gamma - 1));
    p2 = pt2 / (1 + (gamma - 1) / 2 * M2^2)^(gamma / (gamma - 1));

    %% module 3
    gamma = 1.3; % Changed after the combustor
    Tt3_max = 2400;

    % Choking check
    Tt3_choked = Tt2 * (1 / (2 * (gamma + 1))) * ...
        ((1 / (M2^2) * ((1 + gamma * M2^2)^2))) * ...
        ((1 + (((gamma - 1) / 2) * M2^2)))^(-1);

    if Tt3_choked < Tt3_max
        Tt3 = Tt3_choked;
        M3 = 1;
    else
        Tt3 = Tt3_max;
        C = (Tt3 / Tt2) * ((1 + (gamma - 1) / 2 * M2^2) / ((1 + gamma * M2^2)^2)) * M2^2;
        a = C * (gamma^2) - ((gamma - 1) / 2);
        b = 2 * C * gamma - 1;
        c = C;
        M3_roots = roots([a, b, c]);
        M3_squared = M3_roots(M3_roots > 0 & M3_roots < 2);
        M3 = sqrt(M3_squared);
    end

    q23 = 986 * (Tt3 - Tt2) + 0.5 * 0.179 * (Tt3^2 - Tt2^2);
    T3 = Tt3 ./ (1 + (gamma - 1) ./ 2 * M3.^2);
    p3 = p2;
    pt3 = p3 * ((1 + (gamma - 1) / 2 * M3.^2).^(gamma / (gamma - 1)));

    %% module 4
    Ae = 0.015;
```

```
    eta_n = 0.94;
    Tte = Tt3;

    test_M = sqrt(2 / (gamma - 1)) * sqrt((eta_n * ...
        ((1 - (p1 / pt3).^((gamma - 1) / gamma))) / ...
        (1 - eta_n * (1 - (p1 / pt3).^((gamma - 1) / gamma)))));

    if test_M < 1
        Me = test_M;
        pe = p1;
    else
        Me = 1;
        pe = pt3 * (1 - (1 / eta_n) * (gamma - 1) / (gamma + 1))^(gamma / (gamma - 1));
    end

    Te = Tte / (1 + (gamma - 1) / 2 * Me^2);
    ae = sqrt(gamma * R * Te);
    Ve = Me * ae;
    rho_e = pe * 1000 / (R * Te);
    m_dot_e = rho_e * Ve * Ae;

    %% module 6
    g0 = 9.81;
    m_dot_i = m_dot_e / (1 + q23 / 43.2e6);
    m_dot_f = m_dot_e - m_dot_i;
    f = m_dot_f / m_dot_i;
    jet_thrust = m_dot_i .* (1 + f) .* Ve - m_dot_i .* V1;
    pressure_thrust = (pe - p1) * 1000 * Ae;
    total_thrust = jet_thrust + pressure_thrust;
    Veq = Ve + ((pe - p1) * 1000 * Ae / m_dot_e);
    TSFC = (m_dot_f ./ total_thrust) * 3600;
    thermal_efficiency = (m_dot_e .* Veq^2 ./ 2 - m_dot_i .* V1^2 ./ 2) ./ (m_dot_i .* q23);
    propulsive_efficiency = 2 ./ (1 + (Veq ./ V1));
    overall_efficiency = thermal_efficiency * propulsive_efficiency;

    %% Store results
    overall_efficiency_results(idx) = overall_efficiency;
    thrust_results(idx) = total_thrust;
    TSFC_results(idx) = TSFC;
end

%% Plot the results
figure;
plot(M1_range, overall_efficiency_results, 'LineWidth', 2);
xlabel('Mach Number (M1)');
ylabel('Overall Efficiency');
title('Overall Efficiency vs. Mach Number');
grid on;

figure;
plot(M1_range, thrust_results, 'LineWidth', 2);
xlabel('Mach Number (M1)');
```

```
ylabel('Thrust (N)');
title('Thrust vs. Mach Number');
grid on;

figure;
plot(M1_range, TSFC_results, 'LineWidth', 2);
xlabel('Mach Number (M1)');
ylabel('TSFC (kg/hr/N)');
title('TSFC vs. Mach Number');
grid on;
```

Code for part D:

```

clc; clear; clear all;

% Initialize altitude number range
z_range=2000:1:30000;

% Preallocate arrays for results
overall_efficiency_results = zeros(size(z_range));
thrust_results = zeros(size(z_range));
TSFC_results = zeros(size(z_range));

for idx = 1:length(z_range)
    % Set M1 for this iteration
   % M1 = M1_range(idx);

    %% module 1
    % Constants
    gamma = 1.4; % Specific heat ratio
    R = 286.9; % Specific gas constant in J/(kg*K)
    z_star = 8404; % Scale height in meters
    T_s = 288.0; % Standard temperature at sea level in Kelvin
    p_s = 101.3; % Standard pressure at sea level in kPa

    M1=2.4;
    % Input Parameters
    z = z_range(idx);

    % Calculate T and P for each altitude
    if z < 7958 % Within the troposphere
        T1 = T_s * (1 - (((gamma - 1) / gamma) * (z / z_star)));
        p1 = p_s * ((1 - (((gamma - 1) / gamma) * (z / z_star)))^(gamma / (
            gamma - 1)));
    else % In the tropopause
        T1 = 210.0; % Constant temperature in tropopause
        p1 = 33.6 * exp(-(z - 7958) / 6605);
    end

    % Total-to-static relations
    Tt1 = T1 * (1 + (gamma - 1) / 2 * M1^2);
    pt1 = p1 * (1 + (gamma - 1) / 2 * M1^2)^(gamma / (gamma - 1));

    % Sound speed and velocity
```

```
a1 = sqrt(gamma * R * T1);
V1 = M1 * a1;

%% module 2
gamma = 1.4; % Specific heat ratio
eta_d = 0.92; % Inlet/diffuser efficiency
M2 = 0.15;

Tt2 = Tt1;
T2 = Tt2 / (1 + (gamma - 1) / 2 * M2^2);
pt2 = p1 * (1 + (eta_d * (gamma - 1) / 2) * M1^2)^(gamma / (gamma - 1));
p2 = pt2 / (1 + (gamma - 1) / 2 * M2^2)^(gamma / (gamma - 1));

%% module 3
gamma = 1.3; % Changed after the combustor
Tt3_max = 2400;

% Choking check
Tt3_choked = Tt2 * (1 / (2 * (gamma + 1))) * ...
    ((1 / (M2^2) * ((1 + gamma * M2^2)^2))) * ...
    ((1 + (((gamma - 1) / 2) * M2^2)))^(-1);

if Tt3_choked < Tt3_max
    Tt3 = Tt3_choked;
    M3 = 1;
else
    Tt3 = Tt3_max;
    C = (Tt3 / Tt2) * ((1 + (gamma - 1) / 2 * M2^2) / ((1 + gamma * M2^2)^2)) * M2^2;
    a = C * (gamma^2) - ((gamma - 1) / 2);
    b = 2 * C * gamma - 1;
    c = C;
    M3_roots = roots([a, b, c]);
    M3_squared = M3_roots(M3_roots > 0 & M3_roots < 2);
    M3 = sqrt(M3_squared);
end

q23 = 986 * (Tt3 - Tt2) + 0.5 * 0.179 * (Tt3^2 - Tt2^2);
T3 = Tt3 ./ (1 + (gamma - 1) ./ 2 * M3.^2);
p3 = p2;
pt3 = p3 * ((1 + (gamma - 1) / 2 * M3.^2).^(gamma / (gamma - 1)));

%% module 4
Ae = 0.015;
eta_n = 0.94;
Tte = Tt3;

test_M = sqrt(2 / (gamma - 1)) * sqrt((eta_n * ...
    ((1 - (p1 / pt3).^((gamma - 1) / gamma))) / ...
    (1 - eta_n * (1 - (p1 / pt3).^((gamma - 1) / gamma)))));

if test_M < 1
    Me = test_M;
    pe = p1;
```

```
    else
        Me = 1;
        pe = pt3 * (1 - (1 / eta_n) * (gamma - 1) / (gamma + 1))^(gamma / (
            gamma - 1));
    end

    Te = Tte / (1 + (gamma - 1) / 2 * Me^2);
    ae = sqrt(gamma * R * Te);
    Ve = Me * ae;
    rho_e = pe * 1000 / (R * Te);
    m_dot_e = rho_e * Ve * Ae;

    %% module 6
    g0 = 9.81;
    m_dot_i = m_dot_e / (1 + q23 / 43.2e6);
    m_dot_f = m_dot_e - m_dot_i;
    f = m_dot_f / m_dot_i;
    jet_thrust = m_dot_i .* (1 + f) .* Ve - m_dot_i .* V1;
    pressure_thrust = (pe - p1) * 1000 * Ae;
    total_thrust = jet_thrust + pressure_thrust;
    Veq = Ve + ((pe - p1) * 1000 * Ae / m_dot_e);
    TSFC = (m_dot_f ./ total_thrust) * 3600;
    thermal_efficiency = (m_dot_e .* Veq^2 ./ 2 - m_dot_i .* V1^2 ./ 2) ./ (
        m_dot_i .* q23);
    propulsive_efficiency = 2 ./ (1 + (Veq ./ V1));
    overall_efficiency = thermal_efficiency * propulsive_efficiency;

    %% Store results
    overall_efficiency_results(idx) = overall_efficiency;
    thrust_results(idx) = total_thrust;
    TSFC_results(idx) = TSFC;
end

%% Plot the results
figure;
plot(z_range, overall_efficiency_results, 'LineWidth', 2);
xlabel('Altitude');
ylabel('Overall Efficiency');
title('Overall Efficiency vs. Altitude');
grid on;

figure;
plot(z_range, thrust_results, 'LineWidth', 2);
xlabel('Altitude');
ylabel('Thrust (N)');
title('Thrust vs. Altitude');
grid on;

figure;
plot(z_range, TSFC_results, 'LineWidth', 2);
xlabel('Altitude');
ylabel('TSFC (kg/hr/N)');
title('TSFC vs. Altitude');
grid on;
```

Code for part E:

```
clc; clear; clear all;

% Initialize altitude and Mach number ranges
z_range = 2000:500:20000; % Altitudes in meters
M1_range = 0.8:0.1:5.0;    % Flight Mach numbers

% Preallocate arrays for results
optimal_M1_efficiency = zeros(size(z_range));
optimal_M1_TSFC = zeros(size(z_range));
max_efficiency_results = zeros(size(z_range));
min_TSFC_results = zeros(size(z_range));

for z_idx = 1:length(z_range)
    z = z_range(z_idx); % Set altitude for this iteration

    % Initialize temporary storage for results
    efficiency_for_M1 = zeros(size(M1_range));
    TSFC_for_M1 = zeros(size(M1_range));

    for M1_idx = 1:length(M1_range)
        M1 = M1_range(M1_idx); % Set Mach number for this iteration

        %% module 1
        % Constants
        gamma = 1.4; % Specific heat ratio
        R = 286.9; % Specific gas constant in J/(kg*K)
        z_star = 8404; % Scale height in meters
        T_s = 288.0; % Standard temperature at sea level in Kelvin
        p_s = 101.3; % Standard pressure at sea level in kPa

        % Calculate T and P for each altitude
        if z < 7958 % Within the troposphere
            T1 = T_s * (1 - (((gamma - 1) / gamma) * (z / z_star)));
            p1 = p_s * ((1 - (((gamma - 1) / gamma) * (z / z_star)))^(gamma / (gamma - 1)));
        else % In the tropopause
            T1 = 210.0; % Constant temperature in tropopause
            p1 = 33.6 * exp(-(z - 7958) / 6605);
        end

        % Total-to-static relations
        Tt1 = T1 * (1 + (gamma - 1) / 2 * M1^2);
        pt1 = p1 * (1 + (gamma - 1) / 2 * M1^2)^(gamma / (gamma - 1));

        % Sound speed and velocity
        a1 = sqrt(gamma * R * T1);
        V1 = M1 * a1;

        %% module 2
        gamma = 1.4; % Specific heat ratio
        eta_d = 0.92; % Inlet/diffuser efficiency
        M2 = 0.15;

```

```
        Tt2 = Tt1;
        T2 = Tt2 / (1 + (gamma - 1) / 2 * M2^2);
        pt2 = p1 * (1 + (eta_d * (gamma - 1) / 2) * M1^2)^(gamma / (gamma - 1)
            );
        p2 = pt2 / (1 + (gamma - 1) / 2 * M2^2)^(gamma / (gamma - 1));

        %% module 3
        gamma = 1.3; % Changed after the combustor
        Tt3_max = 2400;

        % Choking check
        Tt3_choked = Tt2 * (1 / (2 * (gamma + 1))) * ...
            ((1 / (M2^2) * ((1 + gamma * M2^2)^2))) * ...
            ((1 + (((gamma - 1) / 2) * M2^2)))^(-1);

        if Tt3_choked < Tt3_max
            Tt3 = Tt3_choked;
            M3 = 1;
        else
            Tt3 = Tt3_max;
            C = (Tt3 / Tt2) * ((1 + (gamma - 1) / 2 * M2^2) / ((1 + gamma * M2
                ^2)^2)) * M2^2;
            a = C * (gamma^2) - ((gamma - 1) / 2);
            b = 2 * C * gamma - 1;
            c = C;
            M3_roots = roots([a, b, c]);
            M3_squared = M3_roots(M3_roots > 0 & M3_roots < 1);
            M3 = sqrt(M3_squared);
        end

        q23 = 986 * (Tt3 - Tt2) + 0.5 * 0.179 * (Tt3^2 - Tt2^2);
        T3 = Tt3 ./ (1 + (gamma - 1) ./ 2 * M3.^2);
        p3 = p2;
        pt3 = p3 * ((1 + (gamma - 1) / 2 * M3.^2).^(gamma / (gamma - 1)));

        %% module 4
        Ae = 0.015;
        eta_n = 0.94;
        Tte = Tt3;

        test_M = sqrt(2 / (gamma - 1)) * sqrt((eta_n * ...
            ((1 - (p1 / pt3)^((gamma - 1) / gamma))) / ...
            (1 - eta_n * (1 - (p1 / pt3)^((gamma - 1) / gamma)))));

        if test_M < 1
            Me = test_M;
            pe = p1;
        else
            Me = 1;
            pe = pt3 * (1 - (1 / eta_n) * (gamma - 1) / (gamma + 1))^(gamma /
                (gamma - 1));
        end

        Te = Tte / (1 + (gamma - 1) / 2 * Me^2);
```

```
        ae = sqrt(gamma * R * Te);
        Ve = Me * ae;
        rho_e = pe * 1000 / (R * Te);
        m_dot_e = rho_e * Ve * Ae;

        %% module 6
        g0 = 9.81;
        m_dot_i = m_dot_e / (1 + q23 / 43.2e6);
        m_dot_f = m_dot_e - m_dot_i;
        f = m_dot_f / m_dot_i;
        jet_thrust = m_dot_i .* (1 + f) .* Ve - m_dot_i .* V1;
        pressure_thrust = (pe - p1) * 1000 * Ae;
        total_thrust = jet_thrust + pressure_thrust;
        Veq = Ve + ((pe - p1) * 1000 * Ae / m_dot_e);
        TSFC = (m_dot_f ./ total_thrust) * 3600;
        thermal_efficiency = (m_dot_e .* Veq^2 ./ 2 - m_dot_i .* V1^2 ./ 2) ./
            (m_dot_i .* q23);
        propulsive_efficiency = 2 ./ (1 + (Veq ./ V1));
        overall_efficiency = thermal_efficiency * propulsive_efficiency;

        %% Store temporary results
        efficiency_for_M1(M1_idx) = overall_efficiency;
        TSFC_for_M1(M1_idx) = TSFC;
    end

    %% Find optimal M1 for this altitude
    [max_efficiency, max_eff_idx] = max(efficiency_for_M1);
    [min_TSFC, min_TSFC_idx] = min(TSFC_for_M1);

    % Store optimal results
    optimal_M1_efficiency(z_idx) = M1_range(max_eff_idx);
    optimal_M1_TSFC(z_idx) = M1_range(min_TSFC_idx);
    max_efficiency_results(z_idx) = max_efficiency;
    min_TSFC_results(z_idx) = min_TSFC;
end

%% Plot the results
figure;
plot(z_range, optimal_M1_efficiency, 'LineWidth', 2);
xlabel('Altitude (m)');
ylabel('Optimal Mach Number for Max Efficiency');
title('Optimal Mach Number vs. Altitude for Max Efficiency');
ylim([0 7]); % Set y-axis limits
grid on;

figure;
plot(z_range, optimal_M1_TSFC, 'LineWidth', 2);
xlabel('Altitude (m)');
ylabel('Optimal Mach Number for Min TSFC');
title('Optimal Mach Number vs. Altitude for Min TSFC');
ylim([0 7]); % Set y-axis limits
grid on;
```

Code for part F:

```matlab
clc; clear; clear all;

% Initialize inlet/diffuser efficiency range
eta_d_range = 0.5:0.05:1.0;

% Preallocate arrays for results
overall_efficiency_results = zeros(size(eta_d_range));
thrust_results = zeros(size(eta_d_range));
TSFC_results = zeros(size(eta_d_range));

% Fixed parameters
z = 4300; % Altitude in meters
M1 = 2.4; % Mach number

for idx = 1:length(eta_d_range)
    eta_d = eta_d_range(idx); % Current diffuser efficiency

    %% module 1
    % Constants
    gamma = 1.4; % Specific heat ratio
    R = 286.9; % Specific gas constant in J/(kg*K)
    z_star = 8404; % Scale height in meters
    T_s = 288.0; % Standard temperature at sea level in Kelvin
    p_s = 101.3; % Standard pressure at sea level in kPa

    % Calculate T and P for the given altitude
    if z < 7958 % Within the troposphere
        T1 = T_s * (1 - (((gamma - 1) / gamma) * (z / z_star)));
        p1 = p_s * ((1 - (((gamma - 1) / gamma) * (z / z_star)))^(gamma / (
            gamma - 1)));
    else % In the tropopause
        T1 = 210.0; % Constant temperature in tropopause
        p1 = 33.6 * exp(-(z - 7958) / 6605);
    end

    % Total-to-static relations
    Tt1 = T1 * (1 + (gamma - 1) / 2 * M1^2);
    pt1 = p1 * (1 + (gamma - 1) / 2 * M1^2)^(gamma / (gamma - 1));

    % Sound speed and velocity
    a1 = sqrt(gamma * R * T1);
    V1 = M1 * a1;

    %% module 2
    M2 = 0.15;

    Tt2 = Tt1;
    T2 = Tt2 / (1 + (gamma - 1) / 2 * M2^2);
    pt2 = p1 * (1 + (eta_d * (gamma - 1) / 2) * M1^2)^(gamma / (gamma - 1));
    p2 = pt2 / (1 + (gamma - 1) / 2 * M2^2)^(gamma / (gamma - 1));

    %% module 3
    gamma = 1.3; % Changed after the combustor
    Tt3_max = 2400;
```

```
    % Choking check
    Tt3_choked = Tt2 * (1 / (2 * (gamma + 1))) * ...
        ((1 / (M2^2) * ((1 + gamma * M2^2)^2))) * ...
        ((1 + (((gamma - 1) / 2) * M2^2)))^(-1);

    if Tt3_choked < Tt3_max
        Tt3 = Tt3_choked;
        M3 = 1;
    else
        Tt3 = Tt3_max;
        C = (Tt3 / Tt2) * ((1 + (gamma - 1) / 2 * M2^2) / ((1 + gamma * M2^2)
            ^2)) * M2^2;
        a = C * (gamma^2) - ((gamma - 1) / 2);
        b = 2 * C * gamma - 1;
        c = C;
        M3_roots = roots([a, b, c]);
        M3_squared = M3_roots(M3_roots > 0 & M3_roots < 1);
        M3 = sqrt(M3_squared);
    end

    q23 = 986 * (Tt3 - Tt2) + 0.5 * 0.179 * (Tt3^2 - Tt2^2);
    T3 = Tt3 ./ (1 + (gamma - 1) ./ 2 * M3.^2);
    p3 = p2;
    pt3 = p3 * ((1 + (gamma - 1) / 2 * M3.^2).^(gamma / (gamma - 1)));

    %% module 4
    Ae = 0.015;
    eta_n = 0.94;
    Tte = Tt3;

    test_M = sqrt(2 / (gamma - 1)) * sqrt((eta_n * ...
        ((1 - (p1 / pt3)^((gamma - 1) / gamma))) / ...
        (1 - eta_n * (1 - (p1 / pt3)^((gamma - 1) / gamma)))));

    if test_M < 1
        Me = test_M;
        pe = p1;
    else
        Me = 1;
        pe = pt3 * (1 - (1 / eta_n) * (gamma - 1) / (gamma + 1))^(gamma / (
            gamma - 1));
    end

    Te = Tte / (1 + (gamma - 1) / 2 * Me^2);
    ae = sqrt(gamma * R * Te);
    Ve = Me * ae;
    rho_e = pe * 1000 / (R * Te);
    m_dot_e = rho_e * Ve * Ae;

    %% module 6
    g0 = 9.81;
    m_dot_i = m_dot_e / (1 + q23 / 43.2e6);
    m_dot_f = m_dot_e - m_dot_i;
```

```matlab
    f = m_dot_f / m_dot_i;
    jet_thrust = m_dot_i .* (1 + f) .* Ve - m_dot_i .* V1;
    pressure_thrust = (pe - p1) * 1000 * Ae;
    total_thrust = jet_thrust + pressure_thrust;
    Veq = Ve + ((pe - p1) * 1000 * Ae / m_dot_e);
    TSFC = (m_dot_f ./ total_thrust) * 3600;
    thermal_efficiency = (m_dot_e .* Veq^2 ./ 2 - m_dot_i .* V1^2 ./ 2) ./ (m_dot_i .* q23);
    propulsive_efficiency = 2 ./ (1 + (Veq ./ V1));
    overall_efficiency = thermal_efficiency * propulsive_efficiency;

    %% Store results
    overall_efficiency_results(idx) = overall_efficiency;
    thrust_results(idx) = total_thrust;
    TSFC_results(idx) = TSFC;
end

%% Plot the results
figure;
plot(eta_d_range, overall_efficiency_results, 'LineWidth', 2);
xlabel('Inlet/Diffuser Efficiency (\eta_d)');
ylabel('Overall Efficiency');
title('Overall Efficiency vs. Inlet/Diffuser Efficiency');
grid on;

figure;
plot(eta_d_range, thrust_results, 'LineWidth', 2);
xlabel('Inlet/Diffuser Efficiency (\eta_d)');
ylabel('Thrust (N)');
title('Thrust vs. Inlet/Diffuser Efficiency');
grid on;

figure;
plot(eta_d_range, TSFC_results, 'LineWidth', 2);
xlabel('Inlet/Diffuser Efficiency (\eta_d)');
ylabel('TSFC (kg/hr/N)');
title('TSFC vs. Inlet/Diffuser Efficiency');
grid on;
```

Code for part G:

```matlab
clc; clear; clear all;

% Initialize nozzle efficiency range
eta_n_range = 0.5:0.05:1.0;

% Preallocate arrays for results
overall_efficiency_results = zeros(size(eta_n_range));
thrust_results = zeros(size(eta_n_range));
TSFC_results = zeros(size(eta_n_range));

% Fixed parameters
z = 4300; % Altitude in meters
M1 = 2.4; % Mach number
eta_d = 0.92; % Fixed inlet/diffuser efficiency
```

```matlab

for idx = 1:length(eta_n_range)
    eta_n = eta_n_range(idx); % Current nozzle efficiency

    %% module 1
    % Constants
    gamma = 1.4; % Specific heat ratio
    R = 286.9; % Specific gas constant in J/(kg*K)
    z_star = 8404; % Scale height in meters
    T_s = 288.0; % Standard temperature at sea level in Kelvin
    p_s = 101.3; % Standard pressure at sea level in kPa

    % Calculate T and P for the given altitude
    if z < 7958 % Within the troposphere
        T1 = T_s * (1 - (((gamma - 1) / gamma) * (z / z_star)));
        p1 = p_s * ((1 - (((gamma - 1) / gamma) * (z / z_star)))^(gamma / (gamma - 1)));
    else % In the tropopause
        T1 = 210.0; % Constant temperature in tropopause
        p1 = 33.6 * exp(-(z - 7958) / 6605);
    end

    % Total-to-static relations
    Tt1 = T1 * (1 + (gamma - 1) / 2 * M1^2);
    pt1 = p1 * (1 + (gamma - 1) / 2 * M1^2)^(gamma / (gamma - 1));

    % Sound speed and velocity
    a1 = sqrt(gamma * R * T1);
    V1 = M1 * a1;

    %% module 2
    M2 = 0.15;

    Tt2 = Tt1;
    T2 = Tt2 / (1 + (gamma - 1) / 2 * M2^2);
    pt2 = p1 * (1 + (eta_d * (gamma - 1) / 2) * M1^2)^(gamma / (gamma - 1));
    p2 = pt2 / (1 + (gamma - 1) / 2 * M2^2)^(gamma / (gamma - 1));

    %% module 3
    gamma = 1.3; % Changed after the combustor
    Tt3_max = 2400;

    % Choking check
    Tt3_choked = Tt2 * (1 / (2 * (gamma + 1))) * ...
        ((1 / (M2^2) * ((1 + gamma * M2^2)^2))) * ...
        ((1 + (((gamma - 1) / 2) * M2^2)))^(-1);

    if Tt3_choked < Tt3_max
        Tt3 = Tt3_choked;
        M3 = 1;
    else
        Tt3 = Tt3_max;
        C = (Tt3 / Tt2) * ((1 + (gamma - 1) / 2 * M2^2) / ((1 + gamma * M2^2)^2)) * M2^2;
```

```
        a = C * (gamma^2) - ((gamma - 1) / 2);
        b = 2 * C * gamma - 1;
        c = C;
        M3_roots = roots([a, b, c]);
        M3_squared = M3_roots(M3_roots > 0 & M3_roots < 1);
        M3 = sqrt(M3_squared);
    end

    q23 = 986 * (Tt3 - Tt2) + 0.5 * 0.179 * (Tt3^2 - Tt2^2);
    T3 = Tt3 ./ (1 + (gamma - 1) ./ 2 * M3.^2);
    p3 = p2;
    pt3 = p3 * ((1 + (gamma - 1) / 2 * M3.^2).^(gamma / (gamma - 1)));

    %% module 4
    Ae = 0.015;
    Tte = Tt3;

    test_M = sqrt(2 / (gamma - 1)) * sqrt((eta_n * ...
        ((1 - (p1 / pt3)^((gamma - 1) / gamma))) / ...
        (1 - eta_n * (1 - (p1 / pt3)^((gamma - 1) / gamma)))));

    if test_M < 1
        Me = test_M;
        pe = p1;
    else
        Me = 1;
        pe = pt3 * (1 - (1 / eta_n) * (gamma - 1) / (gamma + 1))^(gamma / (gamma - 1));
    end

    Te = Tte / (1 + (gamma - 1) / 2 * Me^2);
    ae = sqrt(gamma * R * Te);
    Ve = Me * ae;
    rho_e = pe * 1000 / (R * Te);
    m_dot_e = rho_e * Ve * Ae;

    %% module 6
    g0 = 9.81;
    m_dot_i = m_dot_e / (1 + q23 / 43.2e6);
    m_dot_f = m_dot_e - m_dot_i;
    f = m_dot_f / m_dot_i;
    jet_thrust = m_dot_i .* (1 + f) .* Ve - m_dot_i .* V1;
    pressure_thrust = (pe - p1) * 1000 * Ae;
    total_thrust = jet_thrust + pressure_thrust;
    Veq = Ve + ((pe - p1) * 1000 * Ae / m_dot_e);
    TSFC = (m_dot_f ./ total_thrust) * 3600;
    thermal_efficiency = (m_dot_e .* Veq^2 ./ 2 - m_dot_i .* V1^2 ./ 2) ./ (m_dot_i .* q23);
    propulsive_efficiency = 2 ./ (1 + (Veq ./ V1));
    overall_efficiency = thermal_efficiency * propulsive_efficiency;

    %% Store results
    overall_efficiency_results(idx) = overall_efficiency;
    thrust_results(idx) = total_thrust;
```

```
    TSFC_results(idx) = TSFC;
end

%% Plot the results
figure;
plot(eta_n_range, overall_efficiency_results, 'LineWidth', 2);
xlabel('Nozzle Efficiency (\eta_n)');
ylabel('Overall Efficiency');
title('Overall Efficiency vs. Nozzle Efficiency');
grid on;

figure;
plot(eta_n_range, thrust_results, 'LineWidth', 2);
xlabel('Nozzle Efficiency (\eta_n)');
ylabel('Thrust (N)');
title('Thrust vs. Nozzle Efficiency');
grid on;

figure;
plot(eta_n_range, TSFC_results, 'LineWidth', 2);
xlabel('Nozzle Efficiency (\eta_n)');
ylabel('TSFC (kg/hr/N)');
title('TSFC vs. Nozzle Efficiency');
grid on;
```

Code for part H:

```
clc; clear; clear all;

% Initialize Mach number M2 range
M2_range = 0.1:0.1:2.5;

% Preallocate arrays for results
overall_efficiency_results = zeros(size(M2_range));
thrust_results = zeros(size(M2_range));
TSFC_results = zeros(size(M2_range));

% Fixed parameters
z = 4300; % Altitude in meters
M1 = 2.4; % Fixed Mach number at inlet
eta_d = 0.92; % Fixed inlet/diffuser efficiency
eta_n = 0.94; % Fixed nozzle efficiency

for idx = 1:length(M2_range)
    M2 = M2_range(idx); % Current Mach number entering the combustor

    %% module 1
    % Constants
    gamma = 1.4; % Specific heat ratio
    R = 286.9; % Specific gas constant in J/(kg*K)
    z_star = 8404; % Scale height in meters
    T_s = 288.0; % Standard temperature at sea level in Kelvin
    p_s = 101.3; % Standard pressure at sea level in kPa

    % Calculate T and P for the given altitude
```

```
    if z < 7958 % Within the troposphere
        T1 = T_s * (1 - (((gamma - 1) / gamma) * (z / z_star)));
        p1 = p_s * ((1 - (((gamma - 1) / gamma) * (z / z_star)))^(gamma / (gamma - 1)));
    else % In the tropopause
        T1 = 210.0; % Constant temperature in tropopause
        p1 = 33.6 * exp(-(z - 7958) / 6605);
    end

    % Total-to-static relations
    Tt1 = T1 * (1 + (gamma - 1) / 2 * M1^2);
    pt1 = p1 * (1 + (gamma - 1) / 2 * M1^2)^(gamma / (gamma - 1));

    % Sound speed and velocity
    a1 = sqrt(gamma * R * T1);
    V1 = M1 * a1;

    %% module 2
    Tt2 = Tt1;
    T2 = Tt2 / (1 + (gamma - 1) / 2 * M2^2);
    pt2 = p1 * (1 + (eta_d * (gamma - 1) / 2) * M1^2)^(gamma / (gamma - 1));
    p2 = pt2 / (1 + (gamma - 1) / 2 * M2^2)^(gamma / (gamma - 1));

    %% module 3
    gamma = 1.3; % Changed after the combustor
    Tt3_max = 2400;

    % Choking check
    Tt3_choked = Tt2 * (1 / (2 * (gamma + 1))) * ...
        ((1 / (M2^2) * ((1 + gamma * M2^2)^2))) * ...
        ((1 + (((gamma - 1) / 2) * M2^2)))^(-1);

    if Tt3_choked < Tt3_max
        Tt3 = Tt3_choked;
        M3 = 1;
    else
        Tt3 = Tt3_max;
        C = (Tt3 / Tt2) * ((1 + (gamma - 1) / 2 * M2^2) / ((1 + gamma * M2^2)^2)) * M2^2;
        a = C * (gamma^2) - ((gamma - 1) / 2);
        b = 2 * C * gamma - 1;
        c = C;
        M3_roots = roots([a, b, c]);
        if M2<1
        M3_squared = M3_roots(M3_roots > 0 & M3_roots <= 1);
        else
        M3_squared = M3_roots( M3_roots >=1);
        end
        M3 = sqrt(M3_squared);
    end

    q23 = 986 * (Tt3 - Tt2) + 0.5 * 0.179 * (Tt3^2 - Tt2^2);
    T3 = Tt3 ./ (1 + (gamma - 1) ./ 2 * M3.^2);
    p3 = p2;
```

```
    pt3 = p3 * ((1 + (gamma - 1) / 2 * M3.^2).^(gamma / (gamma - 1)));

    %% module 4
    Ae = 0.015;
    Tte = Tt3;

    test_M = sqrt(2 / (gamma - 1)) * sqrt((eta_n * ...
        ((1 - (p1 / pt3)^((gamma - 1) / gamma))) / ...
        (1 - eta_n * (1 - (p1 / pt3)^((gamma - 1) / gamma)))));

    if test_M < 1
        Me = test_M;
        pe = p1;
    else
        Me = 1;
        pe = pt3 * (1 - (1 / eta_n) * (gamma - 1) / (gamma + 1))^(gamma / (
            gamma - 1));
    end

    Te = Tte / (1 + (gamma - 1) / 2 * Me^2);
    ae = sqrt(gamma * R * Te);
    Ve = Me * ae;
    rho_e = pe * 1000 / (R * Te);
    m_dot_e = rho_e * Ve * Ae;

    %% module 6
    g0 = 9.81;
    m_dot_i = m_dot_e / (1 + q23 / 43.2e6);
    m_dot_f = m_dot_e - m_dot_i;
    f = m_dot_f / m_dot_i;
    jet_thrust = m_dot_i .* (1 + f) .* Ve - m_dot_i .* V1;
    pressure_thrust = (pe - p1) * 1000 * Ae;
    total_thrust = jet_thrust + pressure_thrust;
    Veq = Ve + ((pe - p1) * 1000 * Ae / m_dot_e);
    TSFC = (m_dot_f ./ total_thrust) * 3600;
    thermal_efficiency = (m_dot_e .* Veq^2 ./ 2 - m_dot_i .* V1^2 ./ 2) ./ (
        m_dot_i .* q23);
    propulsive_efficiency = 2 ./ (1 + (Veq ./ V1));
    overall_efficiency = thermal_efficiency * propulsive_efficiency;

    %% Store results
    overall_efficiency_results(idx) = overall_efficiency;
    thrust_results(idx) = total_thrust;
    TSFC_results(idx) = TSFC;
end

%% Plot the results
figure;
plot(M2_range, overall_efficiency_results, 'LineWidth', 2);
xlabel('Mach Number (M2)');
ylabel('Overall Efficiency (\eta_o)');
title('Overall Efficiency vs. Mach Number (M2)');
grid on;

```

```
figure;
plot(M2_range, thrust_results, 'LineWidth', 2);
xlabel('Mach Number (M2)');
ylabel('Thrust (N)');
title('Thrust vs. Mach Number (M2)');
grid on;

figure;
plot(M2_range, TSFC_results, 'LineWidth', 2);
xlabel('Mach Number (M2)');
ylabel('TSFC (kg/hr/N)');
title('TSFC vs. Mach Number (M2)');
grid on;
```

Code for part I:

```
clc; clear; clear all;

% Define the range for parametric analysis
M2_range = 0.1:0.01:5; % Diffuser exit Mach number
Tt3_range = 1500:5:2400; % Combustor exit total temperature

% Preallocate results
Thrust = zeros(length(M2_range), length(Tt3_range));
Efficiency = zeros(length(M2_range), length(Tt3_range));

% Loop over M2 and Tt3 to calculate thrust and efficiency
for i = 1:length(M2_range)
    M2 = M2_range(i);
    for j = 1:length(Tt3_range)
        Tt3 = Tt3_range(j);
        %% module 1

% Constants
gamma = 1.4;         % Specific heat ratio
R = 286.9;           % Specific gas constant in J/(kg*K)
z_star = 8404;       % Scale height in meters
T_s = 288.0;         % Standard temperature at sea level in Kelvin
p_s = 101.3;         % Standard pressure at sea level in kPa

% Input Parameters
z = 27400;
M1=5;

% Calculate T and P for each altitude
    if z < 7958  % Within the troposphere
        T1 = T_s * (1 - (((gamma - 1) / gamma) * (z / z_star)));
        p1 = p_s * ((1 - (((gamma - 1) / gamma) * (z / z_star)))^(gamma / (
            gamma - 1)));
    else              % In the tropopause
        T1 = 210.0;  % Constant temperature in tropopause
        p1 = 33.6 * exp(-(z - 7958) / 6605);
    end

% Total-to-static relations
```

```
Tt1 = T1 * (1 + (gamma - 1) / 2 * M1^2);
pt1 = p1 * (1 + (gamma - 1) / 2 * M1^2)^(gamma / (gamma - 1));

% Sound speed and velocity
a1 = sqrt(gamma * R * T1);
V1 = M1 * a1;
%% module 2

% Constants
gamma = 1.4;         % Specific heat ratio
R = 286.9;           % Specific gas constant in J/(kg*K)

% Inputs from Module 1 or given data
%M2 = 0.15;           % Mach number at State 2
%M2=0.40;
eta_d = 0.92;        % Inlet/diffuser efficiency

% Module 2 Calculations
% Total temperature remains constant (no work or heat transfer)
Tt2 = Tt1;

% Compute static temperature at State 2
T2 = Tt2 / (1 + (gamma - 1) / 2 * M2^2);

% Compute total and static pressures
pt2 = p1 * (1 + (eta_d *(gamma - 1) / 2) * M1^2)^(gamma / (gamma - 1));
p2 = pt2 / (1 + (gamma - 1) / 2 * M2^2)^(gamma / (gamma - 1));

% Compute entropy change across the diffuser
cp2 = 1004; % Specific heat at constant pressure (J/(kg*K)) for air
Delta_s_12 = cp2* log(Tt2 / Tt1) - R * log(pt2 / pt1); % Entropy change (J/(kg*K))

% Compute speed of sound and velocity at State 2
a2 = sqrt(gamma * R * T2); % Speed of sound at State 2
V2 = M2 * a2;              % Velocity at State 2

%% module 3

% Constants
gamma = 1.3;                 % Specific heat ratio changed to 1.3 from 1.4
%Tt3_max = 2400;                 % Maximum allowable total temperature at combustor exit (K)


% Module 3 Calculations
% Step 1: Check if the combustor is thermally choked
Tt3_choked = Tt2 * (1 / (2 * (gamma + 1))) * ...
                      ((1 / (M2^2) * ((1 + gamma* M2^2)^2))) * ...
                      ((1 + (((gamma - 1) / 2) * M2^2)))^(-1);

% If thermally choked, adjust the maximum temperature
if Tt3_choked < Tt3
    Tt3 = Tt3_choked;
```

```
    M3=1;
else
    %Tt3 = Tt3;
    % Solve for M3 in the non-choked case
     % Use quadratic equation to solve for M3
     C = (Tt3 / Tt2) * ((1 + (gamma - 1) / 2 * M2^2) / ((1 + gamma * M2^2)^2)) * M2^2;


    % Quadratic coefficients
    a = C * (gamma^2) - ((gamma - 1)/2);
    b = 2 * C * gamma - 1;
    c = C;

   % Solve the quadratic equation
    M3_roots = roots([a, b, c]);

        if M2<1
        M3_squared = M3_roots(M3_roots > 0 & M3_roots <= 1);
        else
        M3_squared = M3_roots( M3_roots >=1);
        end


    % Take the square root to find M3
    M3 = sqrt(M3_squared);
end

% Step 2: Compute heat added (q23)
q23 = 986 * (Tt3 - Tt2)+0.5*0.179*(Tt3^2 - Tt2^2);

% Step 3: Compute static properties at combustor exit
T3 = Tt3 / (1 + (gamma - 1) / 2 * M3.^2);
p3 = p2;  % Static pressure remains the same in constant-pressure combustion
pt3 = p3 * ((1 + (gamma - 1) / 2 * M3.^2).^(gamma/(gamma-1)));


% Step 4: Compute speed of sound and velocity at combustor exit
a3 = sqrt(gamma * R * T3); % Speed of sound at State 3
V3 = M3 * a3;                              % Velocity at State 3

%need cp3 from T3
cp3=986+0.179*T3;

% Step 5: Compute entropy increase across the combustor
Delta_s_23 = cp3* log(Tt3 / Tt2) - R * log(pt3 / pt2);

%entropy chnage 1-3
Delta_s_31 = cp3* log(Tt3 / Tt1) - R * log(pt3 / pt1);

%% module 4

% Inputs
Ae=0.015;
```

```
eta_n=0.94;
Tte = Tt3;                          % Total temperature at nozzle entrance


% Step 1: Compute test Mach number
test_M = sqrt(2 / (gamma - 1)) * sqrt((eta_n * ...
            ((1 - (p1 / pt3)^((gamma - 1) / gamma))) / ...
            (1 - eta_n * (1- (p1 / pt3)^((gamma - 1) / gamma)))));


% Step 2: Check choking condition
if test_M < 1
    % Nozzle is not choked
    Me = test_M;
    pe = p1;  % Exit pressure equals ambient pressure
else
    % Nozzle is choked
    Me = 1;
    pe = pt3 * (1 - (1 / eta_n) * (gamma - 1) / (gamma + 1))^(gamma / (gamma -
         1));
end

% Step 3: Compute static properties at nozzle exit
Te = Tte / (1 + (gamma - 1) / 2 * Me^2); % Static temperature
pte= pe*((1 + (gamma - 1) / 2 * Me^2)^(gamma / (gamma - 1)));

%Compute speed of sound and velocity at nozzle exit
ae = sqrt(gamma * R * Te);                 % Speed of sound
Ve = Me * ae;                              % Velocity

% Step 4: Compute entropy increase
cpe = 986 + 0.179 * Te; % Specific heat at exit (J/(kg*K))
Delta_s_3e = cpe * log(Tte / Tt3) - R * log(pte / pt3);


% Mass flux
rho_e = pe *1000/ (R * Te); % Convert Pe to Pascals if needed
m_dot_e = rho_e * Ve * Ae;

%% module 5
Tt4=Tte;

% Step 2: Apply exit criterion for eta_n_ext
if test_M < 1
     eta_n_ext = 1;
else
     eta_n_ext = test_M.^ (-0.3);
end

% Step 3: Compute T4 (Static Temperature at State 4)
T4 = Tte * (1 - eta_n_ext * (1 - (p1 / pte)^((gamma - 1) / gamma)));

% Step 4: Compute M4 (Mach Number at State 4)
M4 = sqrt((2 / (gamma - 1)) * ((Tt4 / T4) - 1));
```

```matlab


% Step 6: Compute p4 (Static Pressure at State 4)
p4 = p1; % Assume static pressure matches ambient pressure

% Step 7: Compute pt4 (Total Pressure at State 4)
pt4 = p4 * (1 + (gamma - 1) / 2 * M4^2)^(gamma / (gamma - 1));

% Step 8: Compute velocity at State 4
a4 = sqrt(gamma * R * T4);     % Speed of sound at State 4
V4 = M4 * a4;                   % Velocity at State 4

% Step 9: Compute entropy increase across the nozzle
cp4 = 986 + 0.179 * T4;
Delta_s_4e = cp4 * log(Tt4 / Tte) - R * log(pt4 / pte);
Delta_s_41 = Delta_s_12+Delta_s_23+Delta_s_3e+Delta_s_4e;

%% module 6

% Constants
g0 = 9.81;                  % Gravitational acceleration (m/s^2)
q_f = 43.2e6;               % Heating value of fuel (J/kg)


% Step 1: Compute air mass flow rate (mi)
m_dot_i = m_dot_e / (1 + q23 / q_f);

% Step 2: Compute fuel mass flow rate (mf)
m_dot_f = m_dot_e - m_dot_i;

% Step 3: Compute fuel-to-air ratio (f)
f = m_dot_f / m_dot_i;

% Step 4: Compute thrust
jet_thrust = m_dot_i * (1 + f) * Ve - m_dot_i * V1; % Jet thrust
pressure_thrust = (pe - p1)*1000 * Ae;                      % Pressure thrust
total_thrust = jet_thrust + pressure_thrust;                  % Total thrust

% Step 5: Compute equivalent velocity (Veq)
Veq = Ve + ((pe - p1) * 1000*Ae / m_dot_e);            % Equivalent velocity

% Step 6: Compute TSFC
TSFC = (m_dot_f / total_thrust) * 3600;                     % TSFC in (kg/hr)/N

% Step 7: Compute specific impulse
Isp = total_thrust / (m_dot_f * g0);                        % Specific impulse
    (s)

% Step 8: Compute efficiencies
thermal_efficiency = (m_dot_e * Veq^2 / 2 - m_dot_i * V1^2 / 2) / (m_dot_i *
    q23);
propulsive_efficiency = 2 / (1+(Veq /V1));
overall_efficiency = thermal_efficiency * propulsive_efficiency;

```

```
            Thrust(i, j) = total_thrust;
            Efficiency(i, j) = overall_efficiency;
    end
end

% Find optimal values
[max_thrust, idx_thrust] = max(Thrust(:));
[optimal_i_thrust, optimal_j_thrust] = ind2sub(size(Thrust), idx_thrust);
optimal_M2_thrust = M2_range(optimal_i_thrust);
optimal_Tt3_thrust = Tt3_range(optimal_j_thrust);

[max_efficiency, idx_efficiency] = max(Efficiency(:));
[optimal_i_efficiency, optimal_j_efficiency] = ind2sub(size(Efficiency),
    idx_efficiency);
optimal_M2_efficiency = M2_range(optimal_i_efficiency);
optimal_Tt3_efficiency = Tt3_range(optimal_j_efficiency);

% Display results
fprintf('Maximum Thrust: %.2f N at M2 = %.2f, Tt3 = %.2f K\n', max_thrust,
    optimal_M2_thrust, optimal_Tt3_thrust);
fprintf('Maximum Efficiency: %.4f at M2 = %.2f, Tt3 = %.2f K\n',
    max_efficiency, optimal_M2_efficiency, optimal_Tt3_efficiency);
```