```
%% Dan Maguire, Jack Taliercio, Katie Lerond, Kevin Vanderwest
%% AEROSP 225
%% Final Project

clear;
clc;
close all;
format short g;

%% Altitude Sweep
design_height  = 50E3;          % m         Cruise Altitude
min_height = 0E3;               % m
numAltitudes = 100;      % m

Thrust_ALTITUDE = zeros(1,numAltitudes);
I_sp_ALTITUDE = zeros(1,numAltitudes);
T5_ALTITUDE = zeros(1,numAltitudes);
HEIGHTS = linspace(min_height,design_height,numAltitudes);

for index = 1:numAltitudes
    %% Inputs
    M1      = 3.00;         % unitless              Mach #
    height = HEIGHTS(index);
    % Fuel Type: Hydrogen
    q_HV    = 120E6;        % J/kg                  Heating Value
    MW      = 28.9;         % g/mol                 Molecular Weight
    MW_fuel = 2;            % g/mol                 Molecular Weight
    gamma   = 1.4;          % unitless              Specific Heat Ratio
    m_dot   = 100;          % kg/s                  Mass Flow Rate
    T5_max  = 1800;         % K                     Exit Combustor Temperature


    % Requirements:
    % Inlet efficiency > 0.8
    % Thrust > 60E3 N
    % As small as possible

    Ru      = 8.3144598;     % J/(mol*K)            Universal Gas Constant

    % Calculation of Gas Constants
    R_fuel     = Ru/MW_fuel;        % J/(g*K)
    R_fuel     = R_fuel*1000;       % J/(kg*K)
    cp_fuel    = gamma/(gamma-1)*R_fuel; % J/(kg*K)
    cv_fuel    = 1/(gamma-1)*R_fuel;     % J/(kg*K)

    R     = Ru/MW;                  % J/(g*K)
    R     = R*1000;                 % J/(kg*K)
    cp    = gamma/(gamma-1)*R;      % J/(kg*K)
```

```
cv    = 1/(gamma-1)*R;          % J/(kg*K)

w = 2;  % m, design value, width / depth into page


%% Initial State
[T1, ~, p1, rho1] = atmoscoesa(height);
a1 = sqrt(gamma*R*T1);
[~, Trat, prat, rhorat, ~] = flowisentropic(gamma, M1);
T01 = T1/Trat;     p01 = p1/prat;   rho01 = rho1/rhorat;
u1 = M1*a1;
%A1 = m_dot / rho1 / u1;
h1 = cp*T1;
A1 = 6.1250;
height1 = A1 / w;



%% Vectors for plotting along length
xVecByLength = [-10 0];
pVecByLength = [p1 p1];
p0VecByLength = [p01 p01];
TVecByLength = [T1 T1];
T0VecByLength = [T01 T01];
MVecByLength = [M1 M1];
uVecByLength = [u1 u1];
% can back out h and s vectors from the above; no need to keep track
% actually same with quite a few but whatever

%% Inlet
 %disp('Inlet:');


% OS 1
numShocks = 4;
M = zeros(1,numShocks+1);

p = zeros(1,numShocks+1);
p(1) = p1;
p0 = zeros(1,numShocks+1);
p0(1) = p01;
T = zeros(1,numShocks+1);
T(1) = T1;
T0 = zeros(1,numShocks+1);
T0(1) = T01;
rho = zeros(1,numShocks+1);
rho(1) = rho1;
u = zeros(1,numShocks+1);
```

```
u(1) = u1;

M(1) = M1;
B = zeros(1,numShocks);
%theta = zeros(1,numShocks);

theta = 11;     % deg
starting_guess = 30;

for i = 1:numShocks
    if i == 4
        %theta = 11;
        starting_guess = 50;
    end

    B(i) = fzero(@(B) tand(theta) - 2*cotd(B)* ...
        (M(i)^2*(sind(B))^2 - 1) / (M(i)^2*(gamma + cosd(2*B)) + 2), ...
        starting_guess);
    Mn = M(i)*sind(B(i));
    Mn2 = sqrt((Mn^2 + 2/(gamma-1))/((2*gamma/(gamma-1))*Mn^2 - 1));
    M(i+1) = Mn2/sind(B(i)-theta);

    prat = 1 + (2*gamma)/(gamma+1)*(Mn^2 - 1);
    p(i+1) = p(i)*prat;

    rhorat = (gamma+1)*Mn^2/((gamma-1)*Mn^2 + 2);
    rho(i+1) = rho(i)*rhorat;

    Trat = prat/rhorat;
    T(i+1) = T(i)*Trat;

    [~, Trat, prat, ~, ~] = flowisentropic(gamma, M(i+1));
    p0(i+1) = p(i+1)/prat;
    T0(i+1) = T(i+1)/Trat;
    u(i+1) = sqrt(gamma*R*T(i+1)) * M(i+1);
end


%p0(end)/p0(1)

[mach, Trat, prat, rhorat, downstream_mach, ~] = ...
    flownormalshock(gamma, M(end));

%% State 3
M3 = downstream_mach;
T3 = Trat*T(end);  p3 = prat*p(end);   rho3 = rhorat*rho(end);

[~, Trat, prat, rhorat, ~] = flowisentropic(gamma, M3);
```

```
    T03 = T3/Trat;     p03 = p3/prat;   rho03 = rho3/rhorat;
    a3 = sqrt(gamma*R*T3);
    u3 = M3*a3;
    h3 = cp*T3;

    %p03/p01



    %% Inlet Geometry
    x = 0:0.01:13;
    lower_wall_y = x*tand(theta);
    %shock1_y = x*tand(B(1));

    %plot(x,lower_wall_y,'k',x,height1*ones(1,length(x)),'k',x,shock1_y);
    %hold on;
    %x*sind(B(1)) = h
    hit1_x = height1 / tand(B(1));
    % shock 2 m = -tand(B(2)-theta)
    % shock 2 point : hit1_x, h
    % y - h = -tand(B(2)-theta)*(x-hit1_x)
    %shock2_y = height1 - tand(B(2)-theta)*(x - hit1_x);
    %plot(x,shock2_y);

    % shock2_y = hit2_y = tand(theta)*hit2_x = h1 -tand(B(2)-theta)*(hit2_x -↙
hit1_x)
    hit2_x = (height1+tand(B(2)-theta)*hit1_x) / (tand(theta) + tand(B(2)-↙
theta));

    % shock 3 m = sind(B(3))
    % shock 3 point : hit2_x, sind(theta)*hit2_x
    % y - sind(theta)*hit2_x = sind(B(3))*(x-hit2_x)
    %shock3_y = tand(theta)*hit2_x +tand(B(3))*(x - hit2_x);

    %plot(x,shock3_y);


    % shock3_y = hit3_y = h1 = sind(theta)*hit2_x +sind(B(3))*(x - hit2_x)
    hit3_x = (height1-tand(theta)*hit2_x + tand(B(3))*hit2_x) / ...
        (tand(B(3)));

    % shock 4 m = -sind(B(4)-theta)
    % shock 4 point : hit3_x, h1
    % y - h1 = -sind(B(4)-theta)*(x-hit3_x)
    %shock4_y = height1 - tand(B(4)-theta)*(x-hit3_x);

    %plot(x,shock4_y);
```

```
    % shock4_y = hit4_y = sind(theta)*hit4_x = h1 -sind(B(4))*(hit4_x-↙
hit3_x);
    hit4_x = (height1+tand(B(4)-theta)*hit3_x) / (tand(theta) + tand(B(4)-↙
theta));

%     figure(2);
%     hold on;
    % lower wall
    x = 0:0.01:hit4_x;
%     plot(x,x*tand(theta),'k');

    % upper wall
    x = hit1_x:0.01:13;
%     plot(x,height1*ones(1,length(x)),'k');

    % first shock
    x = 0:0.01:hit1_x;
    shock1_y = x*tand(B(1));
%     plot(x,shock1_y);

    % Vectors for plotting along length
    xVecByLength = [xVecByLength, x];
    pVecByLength = [pVecByLength, p(2)*ones(1,length(x))];
    p0VecByLength = [p0VecByLength, p0(2)*ones(1,length(x))];
    TVecByLength = [TVecByLength, T(2)*ones(1,length(x))];
    T0VecByLength = [T0VecByLength, T0(2)*ones(1,length(x))];
    MVecByLength = [MVecByLength, M(2)*ones(1,length(x))];
    uVecByLength = [uVecByLength, u(2)*ones(1,length(x))];

    % second shock
    x = hit1_x:0.01:hit2_x;
    shock2_y = height1 -tand(B(2)-theta)*(x - hit1_x);
%     plot(x,shock2_y);

    % Vectors for plotting along length
    xVecByLength = [xVecByLength, x];
    pVecByLength = [pVecByLength, p(3)*ones(1,length(x))];
    p0VecByLength = [p0VecByLength, p0(3)*ones(1,length(x))];
    TVecByLength = [TVecByLength, T(3)*ones(1,length(x))];
    T0VecByLength = [T0VecByLength, T0(3)*ones(1,length(x))];
    MVecByLength = [MVecByLength, M(3)*ones(1,length(x))];
    uVecByLength = [uVecByLength, u(3)*ones(1,length(x))];

    % third shock
    x = hit2_x:0.01:hit3_x;
    shock3_y = tand(theta)*hit2_x +tand(B(3))*(x - hit2_x);
%     plot(x,shock3_y);
```

```
    % Vectors for plotting along length
    xVecByLength = [xVecByLength, x];
    pVecByLength = [pVecByLength, p(4)*ones(1,length(x))];
    p0VecByLength = [p0VecByLength, p0(4)*ones(1,length(x))];
    TVecByLength = [TVecByLength, T(4)*ones(1,length(x))];
    T0VecByLength = [T0VecByLength, T0(4)*ones(1,length(x))];
    MVecByLength = [MVecByLength, M(4)*ones(1,length(x))];
    uVecByLength = [uVecByLength, u(4)*ones(1,length(x))];

    % fourth shock
    x = hit3_x:0.01:hit4_x;
    shock4_y = height1 -tand(B(4)-theta)*(x-hit3_x);
%     plot(x,shock4_y);

    % Vectors for plotting along length
    xVecByLength = [xVecByLength, x];
    pVecByLength = [pVecByLength, p(5)*ones(1,length(x))];
    p0VecByLength = [p0VecByLength, p0(5)*ones(1,length(x))];
    TVecByLength = [TVecByLength, T(5)*ones(1,length(x))];
    T0VecByLength = [T0VecByLength, T0(5)*ones(1,length(x))];
    MVecByLength = [MVecByLength, M(5)*ones(1,length(x))];
    uVecByLength = [uVecByLength, u(5)*ones(1,length(x))];

    % end of lower wall
    x = hit4_x:0.01:13;
%     plot(x,(tand(theta)*hit4_x)*ones(1,length(x)),'k');

    height3 = height1 - (tand(theta)*hit4_x);

     % Normal Shock
%     plot([hit4_x, hit4_x], [height1-height3, height1]);


    % Vectors for plotting along length
    length_straight = 0.5;
    x_endInlet = hit4_x + length_straight;
    x = hit4_x : 0.01 : x_endInlet;
    xVecByLength = [xVecByLength, x];
    pVecByLength = [pVecByLength, p3*ones(1,length(x))];
    p0VecByLength = [p0VecByLength, p03*ones(1,length(x))];
    TVecByLength = [TVecByLength, T3*ones(1,length(x))];
    T0VecByLength = [T0VecByLength, T03*ones(1,length(x))];
    MVecByLength = [MVecByLength, M3*ones(1,length(x))];
    uVecByLength = [uVecByLength, u3*ones(1,length(x))];



%     % Shading in walls
```

```
%     v2 = [0, 0;
%         hit4_x, height1 - height3;
%         hit4_x + length_straight,  height1 - height3;
%         hit4_x + length_straight, 0;
%         hit1_x, height1
%         hit4_x + length_straight, height1
%         hit4_x + length_straight, height1 + 0.2
%         hit1_x, height1 + 0.2];
%     f2 = [1 2 3 4;
%         5 6 7 8];
%     patch('Faces',f2,'Vertices',v2,'FaceColor','black')
%
%     axis([0 hit4_x + length_straight 0 4]);
%     axis equal;



    A3 = height3*w;



    %% DIFFUSER
    %disp('Diffuser:');
    %A3 = A3; %Starting area of diffuser
    height3 = A3/w; %Starting Diffuser Height
    A4 = 6; %End area of diffuser
    height4 = A4/w; %Diffuser Height

    %Diffuser Length = 3 m

    %Find A*
    [~, ~, ~, ~, arearat] = flowisentropic(gamma, M3);
    a_star = A3/arearat;

    numPoints = 100;
    Aratios = linspace(A3,A4,numPoints) ./ a_star;

    M4Vec(1) = M3;
    p4Vec(1) = p3;
    T4Vec(1) = T3;
    rho4Vec(1) = rho3;

    for i = 2:length(Aratios)
        M4Vec(i) = fzero(@(M) (1/M)*((2/(gamma+1)) ...
            *(1+ ((gamma-1)/2) * M^2))^((gamma+1)/(2*(gamma-1))) - Aratios↙
(i), ...
            [0.01 1]);
```

```
        [~, Trat, prat, rhorat, ~] = flowisentropic(gamma, M4Vec(i));
        p4Vec(i) = p03*prat;
        T4Vec(i) = T03*Trat;
        rho4Vec(i) = rho03*rhorat;
    end

    %% Diffuser Geometry
%     figure(4);
%     length_diffuser = 3;
%     x_endDiffuser = x_endInlet + length_diffuser;
%     x = linspace(x_endInlet, x_endDiffuser, numPoints);
%     height_diffuser = ((height3 - height4).*x)./(x_endInlet -↙
x_endDiffuser) + (height4*x_endInlet - height3*x_endDiffuser)/(x_endInlet -↙
x_endDiffuser);
%     plot(x,height_diffuser);
%     title('Diffuser Internal Height');
%     xlabel('Length along Engine [m]');
%     ylabel('Height of Diffuser [m]');
    %% State 4
    M4 = M4Vec(end);
    T4 = T4Vec(end);
    p4 = p4Vec(end);
    rho4 = rho4Vec(end);
    %T04 = T03;
    %p04 = p03;
    u4Vec = sqrt(gamma.*T4Vec.*R).*M4Vec;
    u4 = u4Vec(end);
    a4 = sqrt(gamma*R*T4);
    h4 = cp*T4;
    [~, Trat, prat, rhorat, ~] = flowisentropic(gamma, M4);
    T04 = T4/Trat;     p04 = p4/prat;   rho04 = rho4/rhorat;

    %massflow = rho(1)*A3*sqrt(gamma*T(1)*R)*M(1)
    %massflow = rho(end)*A4*sqrt(gamma*T(end)*R)*M(end)



    % Vectors for plotting along length
    length_diffuser = 3;
    x_endDiffuser = x_endInlet + length_diffuser;
    x = linspace(x_endInlet, x_endDiffuser, numPoints);
    xVecByLength = [xVecByLength, x];
    pVecByLength = [pVecByLength, p4Vec];
    p0VecByLength = [p0VecByLength, p04*ones(1,length(x))];
    TVecByLength = [TVecByLength, T4Vec];
    T0VecByLength = [T0VecByLength, T04*ones(1,length(x))];
    MVecByLength = [MVecByLength, M4Vec];
    uVecByLength = [uVecByLength, u4Vec];
```

```
    %% Combustor
    %disp('Combustor:');

    length_injector = 1;
    length_flameholder = 1;
    %length_combustor = ???

    m_dot_fuel = 1; %kg/s CHANGE THIS
    fRatio = m_dot_fuel / m_dot;

    T05 = ((fRatio * q_HV) / cp) + T04;

    % INJECTOR

    % Vectors for plotting along length
    x_endInjector = x_endDiffuser + length_injector;
    x = linspace(x_endDiffuser, x_endInjector, numPoints);
    xVecByLength = [xVecByLength, x];
    pVecByLength = [pVecByLength, p4*ones(1,length(x))];
    p0VecByLength = [p0VecByLength, p04*ones(1,length(x))];
    TVecByLength = [TVecByLength, T4*ones(1,length(x))];
    T0VecByLength = [T0VecByLength, T04*ones(1,length(x))];
    MVecByLength = [MVecByLength, M4*ones(1,length(x))];
    uVecByLength = [uVecByLength, u4*ones(1,length(x))];

    % FLAMEHOLDER --- FANNO FLOW
    K=3;
    p04PP_p04 = 1 - gamma*K/2*M4^2*(1 + (gamma-1)/2*M4^2)^(-gamma/(gamma-1));
    p04PP = p04PP_p04 * p04;

    [~, Trat, prat, rhorat, ~, p0rat, ~] = flowfanno(gamma, M4, 'mach');
    Tstar = T4/Trat;   pstar = p4/prat;     rhostar = rho4/rhorat;  p0star =↙
p04/p0rat;

    [M4PP, Trat, prat, rhorat, urat, ~, fanno] = flowfanno(gamma,↙
p04PP/p0star, 'totalpsub');
    T4PP = Trat*Tstar;   p4PP = pstar*prat;   rho4PP = rhorat*rhostar;
    h4PP = cp*T4PP;

    p04PPVec = linspace(p04, p04PP, numPoints);
    for i = 1:length(p04PPVec)
        [M4PPVec(i), Trat, prat, rhorat, urat, ~, fanno] = flowfanno(gamma,↙
p04PPVec(i)/p0star, 'totalpsub');
        T4PPVec(i) = Trat*Tstar;   p4PPVec(i) = pstar*prat;   rho4PPVec(i) =↙
rhorat*rhostar;
        h4PPVec(i) = cp*T4PPVec(i);
    end
```

```
    % Vectors for plotting along length
    x_endFlameholder = x_endInjector + length_flameholder;
    x = linspace(x_endInjector, x_endFlameholder, numPoints);
    xVecByLength = [xVecByLength, x];
    pVecByLength = [pVecByLength, p4PPVec];
    p0VecByLength = [p0VecByLength, p04PPVec];
    TVecByLength = [TVecByLength, T4PPVec];
    T0VecByLength = [T0VecByLength, T04*ones(1,length(x))];
    MVecByLength = [MVecByLength, M4PPVec];
    u4PPVec = M4PPVec .* sqrt(gamma.*R.*T4PPVec);
    uVecByLength = [uVecByLength, u4PPVec];

    % massflow = (rho4P) * A4 * sqrt(gamma*T4P*R)*M4P

    % COMBUSTION CHAMBER --- RAYLEIGH FLOW

    [~, Trat, prat, rhorat, ~, T0rat, p0rat] = flowrayleigh(gamma, M4PP,↙
'mach');
    Tstar = T4PP/Trat;   pstar = p4PP/prat;     rhostar = rho4PP/rhorat;↙
T0star = T04/T0rat;   p0star = p04PP/p0rat;

    u4PP = sqrt(gamma*T4PP * R) * M4PP;

    [M5, Trat, prat, rhorat, ~, ~, p0rat] = flowrayleigh(gamma, T05/T0star,↙
'totaltsub');
    T5 = Trat*Tstar;   p5 = pstar*prat;   rho5 = rhorat*rhostar;    %p05 =↙
p0rat*pstar;

    T5_ALTITUDE(index) = T5;

    T05Vec = linspace(T04, T05, numPoints);
    for i = 1:length(p04PPVec)
        [M5Vec(i), Trat, prat, rhorat, ~, ~, p0rat] = flowrayleigh(gamma,↙
T05Vec(i)/T0star, 'totaltsub');
        T5Vec(i) = Trat*Tstar;   p5Vec(i) = pstar*prat;   rho5Vec(i) =↙
rhorat*rhostar;    %p05Vec(i) = p0rat*pstar;

        [~, Trat, prat, rhorat, ~] = flowisentropic(gamma, M5Vec(i), 'mach');
        T05Vec(i) = T5Vec(i)/Trat;     p05Vec(i) = p5Vec(i)/prat;   rho05Vec↙
(i) = rho5/rhorat;
        a5Vec(i) = sqrt(gamma*R*T5Vec(i));
        u5Vec(i) = M5Vec(i)*a5Vec(i);
        h5Vec(i) = cp*T5Vec(i);
    end

    % Vectors for plotting along length
    length_combustor = 13;          %% CHANGE
```

```
x_endCombustor = x_endFlameholder + length_combustor;
x = linspace(x_endFlameholder, x_endCombustor, numPoints);
xVecByLength = [xVecByLength, x];
pVecByLength = [pVecByLength, p5Vec];
p0VecByLength = [p0VecByLength, p05Vec];
TVecByLength = [TVecByLength, T5Vec];
T0VecByLength = [T0VecByLength, T05Vec];
MVecByLength = [MVecByLength, M5Vec];
uVecByLength = [uVecByLength, u5Vec];
u5 = M5*sqrt(gamma*R*T5);

% is T5 < 1800?

%Mass is conserved
A5 = A4;
%massflow = rho5*M5*sqrt(gamma*R*T5)*A5

%Length of combustor calcs:
pb = p4PP;
Tb = T4PP;

tb = 325 * 10^(-4)*(pb*9.86*10^(-6))^(-1.6)*exp((-8*10^(-4))*Tb);

uav = .5*(u4PP + u5);

Lb = uav * tb;

%% State 5
[Mrat, Trat, prat, rhorat, arearat] = flowisentropic(gamma, M5, 'mach');
T05 = T5/Trat;     p05 = p5/prat;   rho05 = rho5/rhorat;
a5 = sqrt(gamma*R*T5);
u5 = M5*a5;
h5 = cp*T5;




%% Nozzle
%disp('Nozzle:');
At = (1/arearat)*A5;

%% State 6
A6 = At;
T06 = T05;
p06 = p05;
rho06 = rho05;
[M6, Trat, prat, rhorat, ~] = flowisentropic(gamma, 1, 'Mach');
T6 = T06*Trat;  p6 = p06*prat;  rho6 = rho06*rhorat;
```

```
    a6 = sqrt(gamma*R*T6);
    u6 = M6*a6;
    h6 = cp*T6;


    A6Vec = linspace(A5, A6, numPoints);
    for i = 1:length(A6Vec)
        [M6Vec(i), Trat, prat, rhorat, ~] = flowisentropic(gamma, A6Vec(i)↙
/At, 'sub');
        T6Vec(i) = T06*Trat;  p6Vec(i) = p06*prat;  rho6Vec(i) =↙
rho06*rhorat;
        a6Vec(i) = sqrt(gamma*R*T6Vec(i));
        u6Vec(i) = M6Vec(i)*a6Vec(i);
        h6Vec(i) = cp*T6Vec(i);
    end

    % Vectors for plotting along length
    length_nozzle1 = 13;          %% CHANGE
    x_endNozzle1 = x_endCombustor + length_nozzle1;
    x = linspace(x_endCombustor, x_endNozzle1, numPoints);
    xVecByLength = [xVecByLength, x];
    pVecByLength = [pVecByLength, p6Vec];
    p0VecByLength = [p0VecByLength, p06*ones(1,length(x))];
    TVecByLength = [TVecByLength, T6Vec];
    T0VecByLength = [T0VecByLength, T06*ones(1,length(x))];
    MVecByLength = [MVecByLength, M6Vec];
    uVecByLength = [uVecByLength, u6Vec];




    %% State 7
    p7 = p1;
    T07 = T05;
    p07 = p05;
    rho07 = rho05;
    [M7, Trat, prat, rhorat, arearat] = flowisentropic(gamma, p7/p05,↙
'pres');
    T7 = T07*Trat;  p7 = p07*prat;  rho7 = rho07*rhorat;
    A7 = At * arearat;
    a7 = sqrt(gamma*R*T7);
    u7 = M7*a7;
    h7 = cp*T7;

    A7Vec = linspace(A6, A7, numPoints);
    for i = 1:length(A7Vec)
        [M7Vec(i), Trat, prat, rhorat, ~] = flowisentropic(gamma, A7Vec(i)↙
/At, 'sup');
```

```
        T7Vec(i) = T07*Trat;  p7Vec(i) = p07*prat;  rho7Vec(i) =↙
rho07*rhorat;
        a7Vec(i) = sqrt(gamma*R*T7Vec(i));
        u7Vec(i) = M7Vec(i)*a7Vec(i);
        h7Vec(i) = cp*T7Vec(i);
    end

    % Vectors for plotting along length
    length_nozzle2 = 13;          %% CHANGE
    x_endNozzle2 = x_endNozzle1 + length_nozzle2;
    x = linspace(x_endNozzle1, x_endNozzle2, numPoints);
    xVecByLength = [xVecByLength, x];
    pVecByLength = [pVecByLength, p7Vec];
    p0VecByLength = [p0VecByLength, p07*ones(1,length(x))];
    TVecByLength = [TVecByLength, T7Vec];
    T0VecByLength = [T0VecByLength, T07*ones(1,length(x))];
    MVecByLength = [MVecByLength, M7Vec];
    uVecByLength = [uVecByLength, u7Vec];


    Thrust = m_dot * (u7-u1) + (p7 - p1) * A7;
    g = 9.81;   % m/s^2
    I_sp = Thrust / (m_dot_fuel*g);

    %disp('Combustor Temperature [K]');
    %disp(T5);
    %disp('Thrust [kN]');
    %disp(Thrust * 1E-3);
    %disp('Specific Impulse [s]');
    %disp(I_sp);

    Thrust_ALTITUDE(index) = Thrust;
    I_sp_ALTITUDE(index) = I_sp;

     %% Graph Business
%     figure('Position', [50 50 1200 720])
%     hold on;
%
%     % Pressure
%     subplot(2,3,1);
%     plot(xVecByLength, pVecByLength ./ 1000);
%     grid on;
%     xlabel('Length along Engine [m]');
%     ylabel('Pressure [kPa]');
%
%     % Stagnation pressure
%     subplot(2,3,2);
%     plot(xVecByLength, p0VecByLength ./ 1000);
```

```
%     grid on;
%     xlabel('Length along Engine [m]');
%     ylabel('Stagnation Pressure [kPa]');
%     ylim([0, max(p0VecByLength) ./ 1000 * 1.1]);
%
%     % Temperature
%     subplot(2,3,3);
%     plot(xVecByLength, TVecByLength);
%     grid on;
%     xlabel('Length along Engine [m]');
%     ylabel('Temperature [K]');
%     ylim([0, max(TVecByLength) * 1.1]);
%
%     % Stagnation Temperature
%     subplot(2,3,4);
%     plot(xVecByLength, T0VecByLength);
%     grid on;
%     xlabel('Length along Engine [m]');
%     ylabel('Stagnation Temperature [K]');
%     ylim([0, max(T0VecByLength) * 1.1]);
%
%     % Mach Number
%     subplot(2,3,5);
%     plot(xVecByLength, MVecByLength);
%     grid on;
%     xlabel('Length along Engine [m]');
%     ylabel('Mach');
%     ylim([0, max(MVecByLength) * 1.1]);
%
%     % Flow Speed
%     subplot(2,3,6);
%     plot(xVecByLength, uVecByLength);
%     grid on;
%     xlabel('Length along Engine [m]');
%     ylabel('Flow Speed [m/s]');
%     ylim([0, max(uVecByLength) * 1.1]);
%
%
     %% Enthalpy and Entropy
%     hVecByLength = cp.*TVecByLength;
%     delta_s_RVecByLength = (gamma/(gamma-1)).*log(TVecByLength./T1) - ...
%         log(pVecByLength./p1);
%
%     figure();
%     hold on;
%     plot(delta_s_RVecByLength, hVecByLength ./ h1, '-k', 0, 1, 'ob');
%     title('Mollier Diagram');
%     xlabel('Change in Entropy / R [unitless]');
```

```
%     ylabel('Enthalpy normalized by initial state [unitless]');
%     grid on;
%
%     % Adding states
%     plot((gamma/(gamma-1)).*log(T3/T1) - log(p3/p1), h3/h1, 'o');
%     plot((gamma/(gamma-1)).*log(T4/T1) - log(p4/p1), h4/h1, 'o');
%     plot((gamma/(gamma-1)).*log(T4PP/T1) - log(p4PP/p1), h4PP/h1, 'o');
%     plot((gamma/(gamma-1)).*log(T5/T1) - log(p5/p1), h5/h1, 'o');
%     plot((gamma/(gamma-1)).*log(T6/T1) - log(p6/p1), h6/h1, 'o');
%     plot((gamma/(gamma-1)).*log(T7/T1) - log(p7/p1), h7/h1, 'o');
%
%     % Making plot look nice and adding legend
%     ax = gca;
%     xDist = ax.XLim(2) - ax.XLim(1);
%     ax.XLim(1) = ax.XLim(1) - xDist/4;
%     ax.XLim(2) = ax.XLim(2) + xDist/4;
%     yDist = ax.YLim(2) - ax.YLim(1);
%     ax.YLim(1) = ax.YLim(1) - yDist/4;
%     ax.YLim(2) = ax.YLim(2) + yDist/4;
%     ax.YLim(1) = 0;
%     legend('Process', 'State 1', 'State 3', 'State 4', 'State 4''''', ...
%         'State 5', 'State 6', 'State 7', 'location', 'northwest');
end


%% Altitude Plots
figure(1);
plot(HEIGHTS * 1E-3,Thrust_ALTITUDE * 1E-3);
grid on;
title('Thrust Altitude Performance');
xlabel('Altitude [km]');
ylabel('Thrust [kN]');

figure(2);
plot(HEIGHTS * 1E-3,I_sp_ALTITUDE);
grid on;
title('Specific Impulse Altitude Performance');
xlabel('Altitude [km]');
ylabel('I_s_p [s]');

figure(3);
T5_max = T5_max .* ones(1,numAltitudes);
hold on;
plot(HEIGHTS * 1E-3,T5_ALTITUDE);
plot(HEIGHTS * 1E-3,T5_max,'--r');
grid on;
hold off;
title('Combustion Chamber Temperature over Altitude');
```

```
xlabel('Altitude [km]');
ylabel('Combustion Chamber Temperature [K]');
```