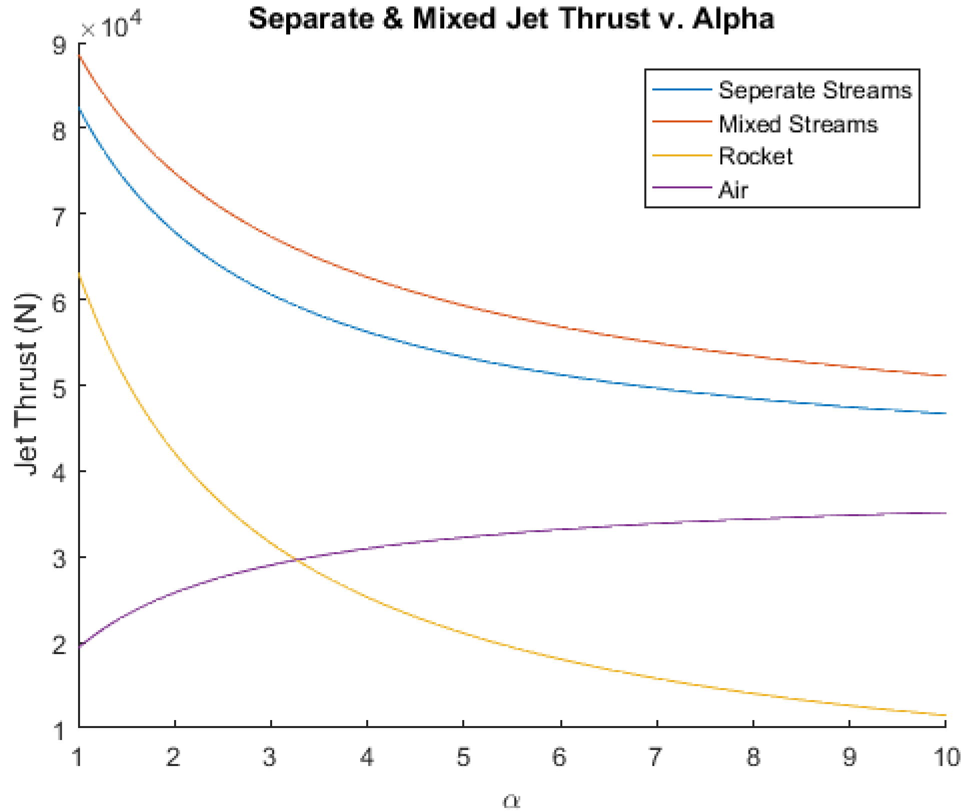
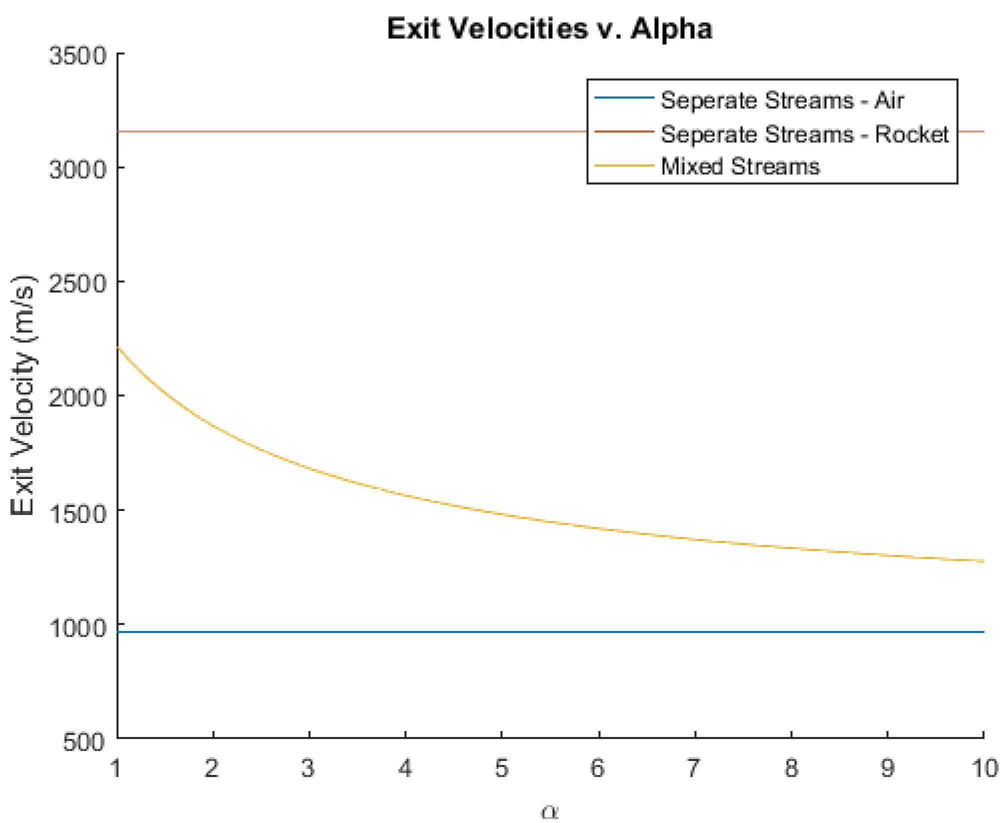
```
% Thomas Satterly
% AAE 537, HW 3
% Problem 2

clear;
close all;

% Define cp curve fits for varios gases
cp_air_f = @(T) 27.453 + 6.1839 * (T / 1000) + 0.89932 * (T /
 1000)^2; % Air CP relation, T in K
cp_co_f = @(T) 27.12 + 6.5565 * (T / 1000) - 0.99939 * (T / 1000)^2;
cp_co2_f = @(T) 75.513 - 0.18732 * T * (10^-3) - 661.85 * (T^(-0.5));
cp_h2_f = @(T) 26.896 + 4.3501 * (T / 1000) - 0.32674 * (T / 1000)^2;
cp_h2o_f = @(T) 29.182 + 14.503 * (T / 1000) - 2.0235 * (T / 1000)^2;

R = 8314; % J / mol*K, Universal gas constant;

% Rocket exit properties
cp_r = 2.0599; %kJ/kg*K
gamma_r = 1.23256;
MW_r = 21.228; % kg/mol
R_r = R / MW_r; % J/mol*K
AeRatio = 5.829; % Area ratio at rocket exit for expansion to match
 static pressure of air
rho_r = 0.14351; % kg/m^3
v_r = 2.904 * 934; % m/s
mdot_r_a = rho_r  * v_r; % Mass flow per meter squared per second,
 kg / m^2 * s
T_r = 1802.63; % K
P_r = 101325; % Pa
M_r = 2.904;

P_r_0 = aeroBox.isoBox.calcStagPressure('mach', M_r, 'gamma',
 gamma_r, 'Ps', P_r);
T_r_0 = aeroBox.isoBox.calcStagTemp('mach', M_r, 'gamma',
 gamma_r, 'Ts', T_r);

% Isentropically expand to 10132.5 Pa
M_r_e = aeroBox.isoBox.machFromPressureRatio('Prat', 10132.5 /
 P_r_0, 'gamma', gamma_r);
T_r_e = aeroBox.isoBox.calcStaticTemp('mach', M_r_e, 'gamma',
 gamma_r, 'Tt', T_r_0);

v_r_e = M_r_e * sqrt(gamma_r * R_r * T_r_e);

% Molecular weights of major species (kg / mol)
MW_co = 28 / 1000;
MW_co2 = 44 / 1000;
MW_h2 = 2 / 1000;
MW_h2o = 18 / 1000;

% Rocket exit major species mass fractions
```

```matlab
mf_co = 3.9048e-1 * MW_co / MW_r * 1000;
mf_co2 = 1.158e-1 * MW_co2 / MW_r * 1000;
mf_h2 = 2.3111e-1 * MW_h2 / MW_r * 1000;
mf_h2o = 2.6242e-1 * MW_h2o / MW_r * 1000;

% Air stream properties


gamma_a = 1.4;
MW_a = 28.97; % g/mol
cp_a = cp_air_f(500) / MW_a * 1000; % J / kg * K
R_a = R / MW_a;
T_a = 500; % K
v_a = 1.5 * sqrt(gamma_a * R_a * T_a);
P_a = 101325; % Pa
rho_a = P_a / (R_a * T_a);
M_a = v_a / sqrt(gamma_a * R_a * T_a);
P_a_0 = aeroBox.isoBox.calcStagPressure('mach', M_a, 'gamma',
 gamma_a, 'Ps', P_a);
T_a_0 = aeroBox.isoBox.calcStagTemp('mach', M_a, 'gamma',
 gamma_a, 'Ts', T_a);

% Isentropically expand to 10132.5 Pa
M_a_e = aeroBox.isoBox.machFromPressureRatio('Prat', 10132.5 /
 P_a_0, 'gamma', gamma_a);
T_a_e = aeroBox.isoBox.calcStaticTemp('mach', M_a_e, 'gamma',
 gamma_a, 'Tt', T_a_0);

v_a_e = M_a_e * sqrt(gamma_a * R_a * T_a_e);

alpha = linspace(1, 10, 100); % Air flow / rocket mass flow
mdot_mix = 40; % kg/s
P_mix = 101325; % Pa
for i = 1:numel(alpha)
    mdot_a = mdot_mix * (alpha(i) / (alpha(i) + 1));
    mdot_r = mdot_mix - mdot_a;

    v_mix = (mdot_a * v_a + mdot_r * v_r) / mdot_mix;

    % Iterate to find CP
    T_mix = (T_a + T_r) / 2; % Starting point
    err = inf;
    T_step = 10;
    lastDir = 1;
    while (abs(err) > 1e-3)
        % Calculate CP first
        cp_a = cp_air_f(T_mix) / MW_a * 1000; % J / kg * K
        cp_co = cp_co_f(T_mix) / MW_co;
        cp_co2 = cp_co2_f(T_mix) / MW_co2;
        cp_h2 = cp_h2_f(T_mix) / MW_h2;
        cp_h2o = cp_h2o_f(T_mix) / MW_h2o;

        % Calculate cp_mix on a mass basis
```

```
        cp_mix = (mdot_r / mdot_mix) * (cp_co * mf_co + cp_co2 *
mf_co2 + cp_h2 * mf_h2 + cp_h2o * mf_h2o) ...
            + (mdot_a / mdot_mix) * cp_a; % J / kg * K

        % Calculate energy balance error
        err = mdot_a * (cp_a * T_a + v_a^2 / 2) + ... % Air stream
            mdot_r * ((cp_r * 1000) * T_r + v_r^2 / 2) - ... % Rocket
stream
            mdot_mix * (cp_mix * T_mix + v_mix^2 / 2); % Mixture
        if sign(err) ~= lastDir
            T_step = T_step / 10;
        end
        lastDir = sign(err);

        if (err > 0)
            T_mix = T_mix + T_step;
        else
            T_mix = T_mix - T_step;
        end
    end
    MW_mix = mdot_mix / ((mdot_a / MW_a) + (mdot_r / MW_r));
    R_mix = R / MW_mix;
    gamma_mix = cp_mix / (cp_mix - R_mix);
    M_mix = v_mix / sqrt(gamma_mix * R_mix * T_mix);
    rho_mix = P_mix / (R_mix * T_mix);
    A_mix = mdot_mix / (v_mix * rho_mix);
    A_a = mdot_a / (v_a * rho_a);
    A_r = mdot_r / (v_r * rho_r);

    % Isentropically expand to 10132.5 Pa

    P_mix_0 = aeroBox.isoBox.calcStagPressure('mach', M_mix, 'gamma',
gamma_mix, 'Ps', P_mix);
    T_mix_0 = aeroBox.isoBox.calcStagTemp('mach', M_mix, 'gamma',
gamma_mix, 'Ts', T_mix);

    % Isentropically expand to 10132.5 Pa
    M_mix_e = aeroBox.isoBox.machFromPressureRatio('Prat', 10132.5 /
P_mix_0, 'gamma', gamma_mix);
    T_mix_e = aeroBox.isoBox.calcStaticTemp('mach', M_mix_e, 'gamma',
gamma_mix, 'Tt', T_mix_0);

    v_mix_e = M_mix_e * sqrt(gamma_mix * R_mix * T_mix_e);

    % Log the results
    results(i).alpha = alpha(i);
    results(i).T_mix = T_mix;
    results(i).M_mix = M_mix;
    results(i).cp_mix = cp_mix;
    results(i).A_mix = A_mix;
    results(i).A_rat = A_mix / (A_a + A_r);
    results(i).sepJetThrust = mdot_a * v_a_e + mdot_r * v_r_e;
    results(i).mixJetThrust = mdot_mix * v_mix_e;
    results(i).r_thrust = mdot_r * v_r_e;
```

```
    results(i).a_thrust = mdot_a * v_a_e;
    results(i).v_a_e = v_a_e;
    results(i).v_r_e = v_r_e;
    results(i).v_mix_e = v_mix_e;
end

figure;
hold on;
xlabel('\alpha');
ylabel('Jet Thrust (N)');
plot([results.alpha], [results.sepJetThrust]);
plot([results.alpha], [results.mixJetThrust]);
plot([results.alpha], [results.r_thrust]);
plot([results.alpha], [results.a_thrust]);
legend('Seperate Streams', 'Mixed Streams', 'Rocket', 'Air');
title('Separate & Mixed Jet Thrust v. Alpha');


figure;
hold on;
xlabel('\alpha');
ylabel('Exit Velocity (m/s)');
plot([results.alpha], [results.v_a_e]);
plot([results.alpha], [results.v_r_e]);
plot([results.alpha], [results.v_mix_e]);
legend('Seperate Streams - Air', 'Seperate Streams - Rocket', 'Mixed
 Streams');
title('Exit Velocities v. Alpha');
```


Separate & Mixed Jet Thrust v. Alpha
×10^4
Jet Thrust (N)
α
Seperate Streams
Mixed Streams
Rocket
Air
9
8
7
6
5
4
3
2
1
1
2
3
4
5
6
7
8
9
10


Exit Velocities v. Alpha
Exit Velocity (m/s)
α
Seperate Streams - Air
Seperate Streams - Rocket
Mixed Streams
3500
3000
2500
2000
1500
1000
500
1
2
3
4
5
6
7
8
9
10

Published with MATLAB® R2016a