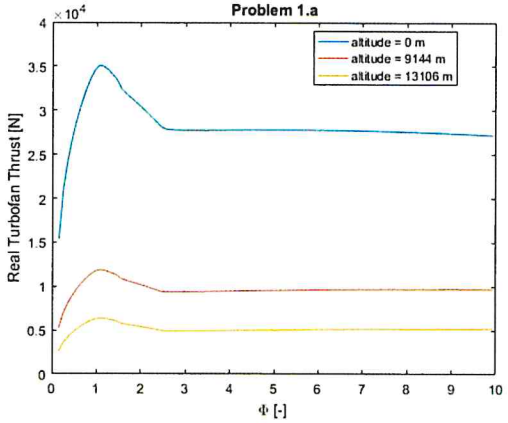
J. Ehlers
jehlers@stanford.edu

Problem Set #3 05/09/2017

1) Turbofan with a Combustion Model

a) → see matlab-Code (Combustor.m)

b) ii) The thrust is optimal for an equivalence ratio of $\Phi \approx 1$ for each altitude. At this equivalence ratio, we achieve the highest flame temperature which leads to the highest thrust. For greater Φ, the momentum of the fuel still increases due to the increasing mass flow rate of the fuel, but the flame temperature would decrease. As the impact of the flame temperature is greater than the impact of the momentum, the maximum thrust is reached at the highest flame temperature.

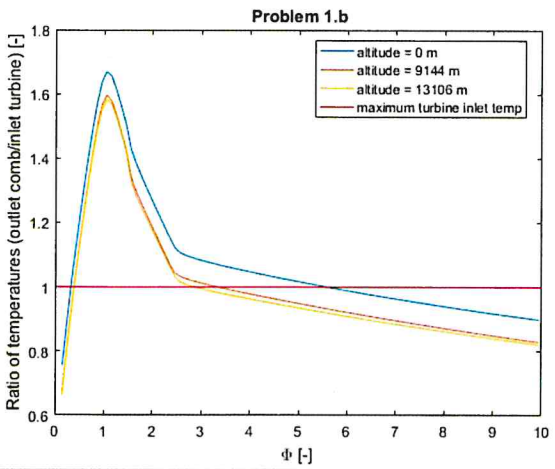


iii) The temperature ratio $\frac{T_4}{T_{4,max}}$ is equal to 1 for the following equivalence ratios:

sea level (0 m): $\Phi_1 = 0.323$ (fuel-lean) $\Phi_2 = 5.598$ (fuel-rich)

cruise cond. (9'144m): $\Phi_1 = 0.386$ $\Phi_2 = 3.357$

cruise cond. (13'106m): $\Phi_1 = 0.390$ $\Phi_2 = 2.893$



We can see from the results above that for $\Phi \approx 0.4$ the thrust is much smaller than the optimal one.
The thrust is approx. 10-20% smaller than at it's peak. This is not optimal.

Nevertheless, to reduce the NO_x production, we need to run our combustor fuel-rich or fuel-lean anyways. We will see this in problem 3.

2) Cruise Conditions for the Honda Jet

a) To analyze the fuel consumption of the engine, we will now assume that the equivalence ratio ϕ stays constant for each altitude at it's optimal value computed in Problem 1b.
From the previous modeling of the real turbofan we can thus keep all quantities that are not dependent on the overall mass flow of air and the weight of the airplane

constant: state variables (T_{t-8}, p_{t-8}, ρ_{t-8}), equivalence ratio ϕ, engine parameters, fluid parameters, air frame parameters, incoming gas properties (U_{inf}, altitude)

variable: $\dot{m}_{air}$ → $\dot{m}_{fuel}$ → thrust available

→ air plane weight → thrust required

Thus, to match thrust (required) with the thrust (available), we have to recompute the thrust (available) for every air mass flow $\dot{m}_{air}$ and then find from $\dot{m}_{air}$ and ϕ the required $\dot{m}_{fuel}$.
Then, for every time step, we have to recompute the required thrust from the new airplane weight and rematch the available thrust to the required thrust by adjusting $\dot{m}_{air}$.

b) → see matlab code: function "match Thrust.m"

- This function computes for the current weight the required thrust: $T_{req} = \frac{c_{D,0}}{2} \cdot \rho \cdot U_\infty^2 \cdot S + \frac{W^2 \cdot 2}{\rho \cdot U_\infty^2 \cdot \pi e b^2}$
- Then, we compute for the current weight the turbofan thrust from 2 engines using the matlab code "realturbofanThrust.m" that we developed before and that we adjusted in this problem set by including a real combustor. ($T_{turbofan,opt} \sim 6'000N$ for $\phi \sim 1$)
- Then, we matched the two thrusts $T_{req.} \approx T_{turbofan}$ by adjusting $\dot{m}_{air}$ for constant ϕ

→ From ϕ and the given total mass flow of air we could then determine the fuel mass flow rate needed to produce the required amount of thrust:
$\dot{m}_{fuel} = 2 \cdot (\dot{m}_{total} - \dot{m}_{air})$ (for two engines).

c)

$$\text{ODE: } \frac{dW_{aircraft}}{dt} = -\dot{m}_{fuel} \qquad \text{with } \dot{m}_{fuel} = \text{fuel mass flow rate}$$

The fuel mass flow rate depends on the equivalence ratio ϕ and the mass flow rate of air, as well as the stoichometric fuel air ratio f_{st}: $\dot{m}_{fuel} = \phi \cdot f_{st} \cdot \dot{m}_{air}$
Thus, for our problem we keep $\phi \approx 0.39 \approx$ const. and adjust $\dot{m}_{air}$ to match the required thrust of the airplane: $\dot{m}_{air} = \dot{m}_{air}(T_{req}(t), T_{real\ turbofan}(t))$ and $\dot{m}_{fuel} = \dot{m}_{fuel}(\dot{m}_{air})$

Solve ODE: $W_{aircraft} = \frac{MTOW}{g} \int_0^t \dot{m}_{fuel}(\tilde{t})\, d\tilde{t}$ with MTOW = max. take off weight

with $\dot{m}_{fuel}(\tilde{t}) = \dot{m}_{fuel}(\dot{m}_{air}(\tilde{t}))$ at $U_{inf} = 150\,m/s$ and altitude = 30,000ft

(For this cruise condition, we chose $\phi = 0.398$ as computed in Problem 1 for this altitude and found flow rates over time as: $\dot{m}_{air} \in [0.6394, 0.5586]\,kg/s$
$\dot{m}_{fuel} \in [0.0331, 0.0290]\,kg/s$

d) The fuel mass flow needed to produce the required amount of thrust is very small ($T_{req} \approx 1700N$) compared to the available turbofan thrust of 2 engines ($T_{fan} \sim 8000\,N$)
Thus, the air mass flow needed to produce the thrust is very small compared to $\dot{m}_{air,max} = 2.99\,kg/s$ at an altitude of 30,000ft.
Thus, assuming all fuel is consumed with an initial fuel weight of $W_{fuel} = 5'600N = g \cdot 571\,kg$ we computed a range of approx. 3720km in 310min ≈ 6h

e) Using the Breguet range equation, we obtained a range between 2'810 and 3'590km which is smaller than our computed range using the real turbofan model with a real combustor.

This can be explained by the computation of the fuel consumption. While the Breguet range equation worked with a given (measured) thrust specific fuel consumption, we determined the thrust specific fuel consumption using our combustor model for lean combustion. The given TSFC was for a flight speed of $Ma = 0.7$, which leads to $U_{inf} = 212\,m/s$* at an altitude of 30'000ft. Thus, the range with the Breguet equation was computed for a much higher flight velocity which, of course, needs more fuel and thus leads to a smaller range. (We computed the range for a velocity of $U_{inf} = 150\,m/s$)
Both models (Breguet range equation and our model) neglect the effects of compressibility drag, worked with an optimistic lift over drag ratio, and neglected the fuel to climb and to decrease in range while take-off and landing.
Thus, both ranges are an overestimation.

* $U_{inf} = Ma \cdot a(altitude)$

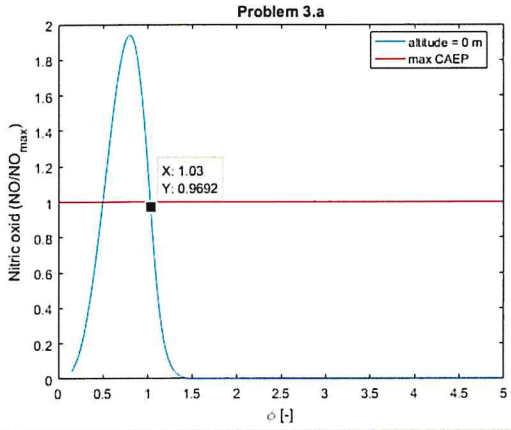
Problem #3) Nitric Oxid Formation

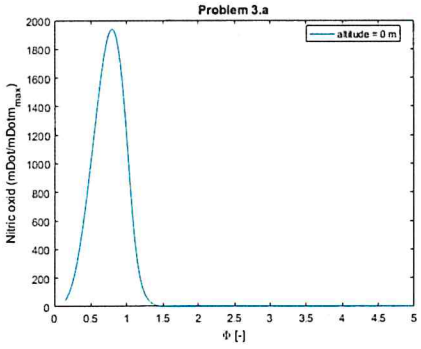
a) → Compute mass fraction of NO using cantera (see matlab code)

- altitude: 0ft. , $U_\infty = U_0 = 90\ m/s$

↳ Plot $\dot{m}_{NO} / \dot{m}_{NO,max}$ over $\phi \in [0.15, 5.0]$ with $\dot{m}_{NO,max} = 0.04\ g/s$

→ $\dot{m}_{NO} = X_{NO} \cdot \dot{m}_{all} = X_{NO} \cdot \dot{m}_{air} (1+f)$





$\dot{m}_{NO}\ [kg/s]\ /1000$

We can observe from the plot that the CAEP emission standards are only met for an equivalence ratio above $\phi \sim 1$. At this condition, the fuel consumption is much higher than the minimum fuel consumption, that we need for the required thrust ($\phi \sim 0.4$). Furthermore, at $\phi \approx 1$ the flame temperature is still too high, such that we need an even higher equivalence ratio of $\phi > 5.6$ for the sea level. This would increase the fuel consumption even more.

b) RQL-combustion strategy:

(i)

production rate of NO : $\frac{d\dot{m}_{NO}}{dt} = k \cdot \dot{m}_{N_2} \cdot \dot{m}_O \ [kg/s^2] \approx CTE \rightarrow \dot{m}_{NO} = \frac{d\dot{m}_{NO}}{dt} \cdot \tau_{dil} \overset{!}{\leq} \dot{m}_{NO,max}$

↳ $\tau_{dil} = \frac{L_{dil}}{U_4} \quad \leftrightarrow \quad \dot{m}_{NO,max} \geq \frac{L_{dil}}{U_4} \cdot k \cdot \dot{m}_{N_2} \cdot \dot{m}_O \quad \leftrightarrow \quad \boxed{L_{dil} \leq \frac{U_4 \cdot \dot{m}_{NO,max}}{k \cdot \dot{m}_{N_2} \cdot \dot{m}_O}}$

with $U_4 = 11.8\ m/s$, $\dot{m}_{N_2} = 0.403\ \frac{kg}{s}$, $\dot{m}_O = 3.15 \times 10^{-7}\ \frac{kg}{s}$, $k = 2.66\ \frac{1}{kg}$

and $\dot{m}_{NO,max} = 0.04 \times 10^{-3}\ kg/s$

we can compute $L_{dil,max} = \underline{1'398\ m}$

↳ Yes, based on this result it seems very feasible to dilute the burned gas before significant NO is produced.

(ii) 1) Rich burn:

- Compute T_3, p_3 from engine
- Compute primary mass flow of air: $\beta_c = \frac{\dot{m}_{a2}}{\dot{m}_{a1}}$ with $\dot{m}_{a2} + \dot{m}_{a1} = \dot{m}_a$ (total air mass flow) ($\dot{m}_{a1}$)

$$\rightarrow \dot{m}_{a1} = \dot{m}_a - \dot{m}_{a2} = \dot{m}_a - \dot{m}_{a1}\beta_c \rightarrow \dot{m}_{a1} = \dot{m}_a \cdot \frac{1}{1+\beta_c}$$

- Compute mass flow of fuel: $\dot{m}_{fuel} = \dot{m}_a \cdot f = \dot{m}_a \cdot \phi \cdot f_{st}$ ($\dot{m}_{fuel}$)
- Compute equilibrium between $\dot{m}_{a1}$ and $\dot{m}_{fuel}$ → rich burn

2) Quenching:

- Eliminate NO from products from rich burn

Dilution:

- Mixing with reworked, uncombusted air:

$T_{a2} = T_3$	$T_{a1} = T_{flame}$
$\dot{m}_{a2}$	$\dot{m}_{a1} + \dot{m}_{fuel}$
c_{pa2}	c_{pa1}

$$T_{dil} = \frac{1}{\dot{m}_a + \dot{m}_{fuel}} \cdot \frac{1}{c_{p,dil}} \left(c_{p1}(\dot{m}_{a1} + \dot{m}_{fuel}) T_{flame} + c_{p2}\, \dot{m}_{a2}\, T_3 \right)$$

$p_{dil} = p_3$

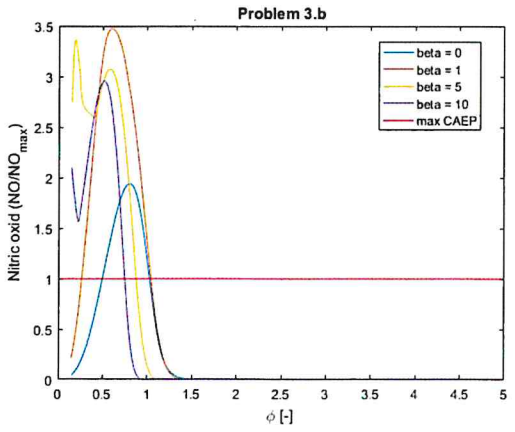
$\dot{m}_{dil} = \dot{m}_{a1} + \dot{m}_{a2} + \dot{m}_{fuel}$ [get mass fraction as $(\dot{m}_{a1} + \dot{m}_{fuel}) \cdot y_{products} + \dot{m}_{a2} \cdot y_{air}$ ← mass fractions]

3) Lean burn:

- Compute equilibrium (HT with cantera)

⇒ Compute NO mass flow: $\dot{m}_{NO} = X_{NO} \cdot \dot{m}_{dil}$ with $\dot{m}_{dil} = \dot{m}_{a1} + \dot{m}_{a2} + \dot{m}_{fuel}$ for $\dot{m}_{a1} + \dot{m}_{a2} = 8\ kg/s$

→ Compute ratio $\dot{m}_{NO} / \dot{m}_{NO,max}$ and plot it over equivalence ratio ϕ:



Thus, using RQL-design, emissions are still the limiting factor of the engine performance for low fuel-air ratios.

Depending on β, only equivalence ratios

$\phi < 0.5$, $\phi > 1$	for $\beta = 0$
$\phi < 0.2$, $\phi > 1$	for $\beta = 1$
$\phi > 0.9$	for $\beta = 5$
$\phi > 0.7$	for $\beta = 10$

are within the emission standards of $\dot{m}_{NO,max} = 0.04\ g/s$.

→

We can observe, that an increasing mass flow ratio β leads to a more fuel-lean second combustion and therefore to a shift of the NO-production curve to the left. Moreover, the higher the mass of air ($\dot{m}_{a1}$) is in the first combustion zone (fuel rich), the higher is the temperature after the mixing with the reroutet air ($\dot{m}_{a2}$ is smaller). Therefore, we have a higher maximum NO-production for smaller β, as the NO production is highly dependent on the gas temperature and the temperature in the second zone is higher for small β.

c) Empirical correlations: $X_{NO_x} = 3.32 \times 10^{-12} \exp(0.008 T_4) p_4^{0.4}$ $U_0 = 90\,m/s$

We know from problem 1 that for $\phi \approx 0.3228$: $T_4 = 1600\,K$, $p_4 = 25.468\,bar$

$\rightarrow X_{NO} \approx X_{NO_x} = 4.3904 \times 10^{-4}$

We also assume $\dot{m}_{air} = 8\,kg/s$ for take off condition and a negligible fuel mass flow rate compared to the air mass flow.

Thus, $\dot{m}_{NO} = n_{NO} \cdot MW_{NO}$ $\rightarrow \dot{m}_{NO} = MW_{NO} \cdot X_{NO} \cdot \dot{n}_m = MW_{NO} \cdot X_{NO} \cdot \frac{\dot{m}_m}{MW_m} = MW_{NO} \cdot X_{NO} \cdot \frac{\dot{m}_m}{\sum x_i MW_i}$

Now, we know that $MW_{NO} = 30\,g/mol$. Thus, for every mol of exhaust gases, we release ($MW_{NO} \cdot X_{NO_x} = 0.0132\,g/n_m$) $0.0132\,g$ of NO.

For air, we can compute $MW_m = \sum_{i=1}^{5} x_i MW_i = 28.789\,g/mol$.

$$\rightarrow \dot{n}_m = \frac{\dot{m}_m}{MW_m} = \frac{\dot{m}_{air} + \dot{m}_{fuel}}{MW_m} = \frac{8.1731\,kg/s}{28.7888\,g/mol} = 283.9\,mol/s$$

Thus, we can compute $\dot{m}_{NO} = MW_{NO} \cdot X_{NO} \cdot \dot{n}_m = 3.7\,g/s$

This model shows again, that the Honda Jet would not meet the CAEP emission standards at it's maximum pollution. But as shown in part 3b), with a different equivalence ratio ϕ, the maximum emission of $NO_{max} = 0.04\,g/s$ can be met for $\phi \neq \phi_{opt}$.

ME 257: Homework 3

Johanna Ehlers - 06111975 - jehlers@stanford.edu

HW 3 - Matlab Code

1 Problem 1

1.1 Combustor.m

```
function [p4,T4,mDot4] = combustor(phi,UInf, altitude, engine, fluid)
    % function: combustor
    % ------------------------------------------------------------------
    % This function returns the turbine inlet temperature and mass
    % fractions.
    %   Inputs:
    %       phi = equivalence ratio
    %       UInf = free stream velocity [m/s]
    %       engine = structure containing the engine parameters
    %       fluid = structure containing the fluid properties including the
    %           cantera solution object
    %   Outputs:
    %       T4 = The turbine inlet temperature [K]
    %       mDot4 = composition of the turbine inlet gas (in mass flow) [kg/s]
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %TODO:  Calculate the temperature (T4) and outlet mass flow rates from
    %       the combustor (mDot4). Note that any changes to the cantera gas
    %       object fluid.gas remain even after function calls. The
    %       combustion should occur via an HP reactor. That is use
    %           equilibrate(fluid.gas,'HP')
    %       where fluid.gas is the cantera phase object at T3,p3,X3.
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % Given states
    [s0.T, s0.a, s0.p, s0.r]=atmosisa(altitude);
    [~, ~, ~, sSL.r]=atmosisa(0);

    % Given mass flow rate:
    dmdt=engine.mDotSeaLevel*s0.r/sSL.r;


    % Compute combustion temperature
    setCombustorInletConditions(phi,UInf, altitude, engine, fluid);
    f = fuelAirRatio(fluid);

    equilibrate(fluid.gas,'HP');
    T4=temperature(fluid.gas);
    p4=pressure(fluid.gas);

    % Compute mass flow rate
    mDot4=dmdt.*(1+f);
```



```
end
function setCombustorInletConditions(phi,UInf, altitude, engine, fluid)
    % function: setCombustorInletConditions
    % ------------------------------------------------------------------
    % This function sets the fluid.gas solution object to the combustor
    % inlet conditions after the fuel is mixed with the oxidizer
    %   Inputs:
    %       phi = equivalence ratio
    %       UInf = free stream velocity [m/s]
    %       engine = structure containing the engine parameters
    %       fluid = structure containing the fluid properties including the
    %           cantera solution object
    %   Outputs:
    %       fluid.gas object is altered to inlet state
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %TODO:  Calculate the inlet conditions to the combustor:
    %       T3,p3
    %       The provided code sets the properties of the inlet gas. Note
    %       that the fluid object is assumed to have additional fields:
    %       fluid.gas = cantera gas object
    %       fluid.fuel.name = fuel name in cti file (e.g., 'nc12h26')
    %       fluid.fuel.nC = #number of hydrogen atoms in fuel (e.g., 12)
    %       fluid.fuel.nH = #number of hydrogen atoms in fuel (e.g., 26)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %Compute the inlet gas condition
    % Given states
    [s0.T, s0.a, s0.p, s0.r]=atmosisa(altitude);
    [~, ~, ~, sSL.r]=atmosisa(0);
    s0.u=UInf;
    s0.M=UInf./s0.a;

    % Given fluid
    air.R=fluid.R;
    air.g=fluid.gamma;
    air.cp=air.R* air.g/(air.g-1);

    % Given pressure ratios:
    ratio_c=engine.overallCompressionRatio;

    % Given efficiencies:
    e.diff=engine.efficiencies.diffuser;
    e.fan=engine.efficiencies.fan;
    e.cop=engine.efficiencies.compressor;

    % Solve Diffusor
    s2.T=s0.T.*( 1 +(air.g-1)/2 *s0.M.^2 );
    s2.p=s0.p.*( 1+ e.diff*(s2.T./s0.T-1)) .^(air.g/(air.g-1)) ;

    % Solve compressor
    s3.p=ratio_c.*s2.p;
    s3.T=s2.T.*(1+ 1/e.cop/e.fan *(ratio_c^((air.g-1)/air.g) -1) );

    T3=s3.T;
    p3=s3.p;

    %set the inlet gas
    nSp = nSpecies(fluid.gas);
    X3 = zeros(nSp,1);
```

```
    iFuel = speciesIndex(fluid.gas,fluid.fuel.name);
    X3(iFuel)=phi;
    iO2 = speciesIndex(fluid.gas,'o2');
    iN2 = speciesIndex(fluid.gas,'n2');
    X3(iO2)= fluid.fuel.nC+fluid.fuel.nH/4.0;
    X3(iN2)=3.76*X3(iO2);
    set(fluid.gas,'T',T3,'P',p3,'X',X3)
end

function f = fuelAirRatio(fluid)
    % function: fuelAirRatio
    % -------------------------------------------------------------
    % Determines the fuel-air ratio of the unburned gas.
    %   Inputs:
    %       fluid = structure containing the fluid properties including the
    %           cantera solution object
    %   Outputs:
    %       f = fuel-air ratio
    Y = massFractions(fluid.gas);
    iFuel = speciesIndex(fluid.gas,'nC12H26'); %fluid.fuel.name);
    iO2 = speciesIndex(fluid.gas,'o2');
    iN2 = speciesIndex(fluid.gas,'n2');
    f = Y(iFuel)/(Y(iO2)+Y(iN2));
end
```

1.2 matchTemperature.m

```
function [s_phi_up,s_phi_low] = matchTemperature(UInf,altitude,engine,fluid)
% Find equivalence ratio phi for outlet temperature of T=1600
maxIt=10000;
T4_max=1600;

iteration=0;
phi_ratio=1;
T4=2*T4_max;
while (iteration<maxIt && T4>T4_max)
    [~,s4,~]=realTurbofanThrust(phi_ratio,UInf,altitude,engine,fluid);
    T4=s4.T;
    error_rel=(T4-T4_max)/T4_max;

    phi_ratio=phi_ratio+0.001;
    tolerance=abs(error_rel)*100;
    iteration=iteration+1;
end
s_phi_up=phi_ratio;

phi_ratio=1;
iteration=0;
T4=2*T4_max;
while (iteration<maxIt && T4>T4_max)
    [~,s4,~]=realTurbofanThrust(phi_ratio,UInf,altitude,engine,fluid);
    T4=s4.T;
    error_rel=(T4-T4_max)/T4_max;

    phi_ratio=phi_ratio-0.001;
    tolerance=abs(error_rel)*100;
    iteration=iteration+1;
end
s_phi_low=phi_ratio;
```


1.3 Plot of thrust and outlet temperature over equivalence ratio

```
function plotTemp4(phi_ratio,T4,altitudes)
    % This function builds the plots required for problem 1b of problem set
    % 3.
    %   Inputs:
    %       UInf = vector of freestream velocities. [m/s]
    %       phi_ratio = array of equivalence ratio. [-]
    %       altitudes=vector of altitudes [m].
    %           T4=vector of outlet temperature [K].
    %   Outputs:
    %       plots of outlet temperature vs equivalence ratio for the inputed
        altitudes.

    nAltitudes = length(altitudes);

    %this loop creates the strings used for the legend
    legendEntries = cell(1, nAltitudes);
    for i = 1:nAltitudes
        legendEntries{1,i} = ['altitude = ' num2str(altitudes(i)) ' m'];
    end

    %create the plots
    figure()
    plot(phi_ratio, T4./1600);
    hold on;
    plot([0,10],[1,1],'r');
    hold on
    title('Problem 1.b')
    xlabel('\Phi [-]')
    ylabel('Ratio of temperatures (outlet comb/inlet turbine) [-]')
    legend([legendEntries,'maximum turbine inlet temp'],'location','best')
end

function plotThrust(phi_ratio,Ts,altitudes)
    % function: problemSet1_problem1a
    % -------------------------------------------------------------
    % This function builds the plots required for problem 1a of problem set
    % 1.
    %   Inputs:
    %       UInf = vector of freestream velocities. [m/s]
    %       Ts = array of required thrust at different altitudes. [N]
    %       altitudes=vector of altitudes [m].
    %   Outputs:
    %       plots of required thrust vs UInf for the inputed altitudes.

    nAltitudes = length(altitudes);

    %this loop creates the strings used for the legend
    legendEntries = cell(1, nAltitudes);
    for i = 1:nAltitudes
        legendEntries{1,i} = ['altitude = ' num2str(altitudes(i)) ' m'];
    end

    %create the plots
    figure()
    plot(phi_ratio, Ts)
    hold on
    title('Problem 1.a')
```

```
57      xlabel('\Phi [-]')
58      ylabel('Real Turbofan Thrust [N]')
59      legend(legendEntries,'location','best')
60
61  end
```


2 Problem 2

2.1 Driver - Problem 2

```
1  % Script: problemSet3Driver
2  %-------------------------------------------------------------------------
3  %   This script is used as the driver for problem set one. Feel free to
4  %   tinker with the parameters to get a feel for their sensitivities, but
5  %   the final submission must use the parameters as is.
6
7  clear; close all; clc
8
9  %define a structure containing the HondaJet's parameters
10 HondaJet.airframe.S = 17.3;                          %[m^2], reference area
11 HondaJet.airframe.b = 12.12;                         %[m], wing span
12 HondaJet.airframe.W = 41000;                         %[N], weight (which is max
       takeoff weight)
13 HondaJet.airframe.CD0 = 0.01;                        %[], zero-lift drag
       coefficient (or parasite drag coeff.)
14 HondaJet.airframe.e = 0.8;                           %[], Oswald Efficiency
15 HondaJet.engine.overallCompressionRatio = 24;        %[]
16 HondaJet.engine.turbineInletTemperature=1600;        %[K]
17 HondaJet.engine.mDotSeaLevel = 8.0;                  %[kg/s]
18 HondaJet.engine.nEngines = 2;                        %[]
19 HondaJet.engine.bypassRatio = 2.9;                   %[]
20 HondaJet.engine.fanCompressionRatio = 2.0;           %[]
21 HondaJet.engine.efficiencies.diffuser = 0.95;        %[]
22 HondaJet.engine.efficiencies.fan = 0.92;             %[]
23 HondaJet.engine.efficiencies.compressor = 0.87;      %[]
24 HondaJet.engine.efficiencies.combustor = 1.00;       %[]
25 HondaJet.engine.efficiencies.turbine = 0.91;         %[]
26 HondaJet.engine.efficiencies.coreNozzle = 0.98;      %[]
27 HondaJet.engine.efficiencies.fanNozzle = 0.98;       %[]
28
29 %define a structure containing the fluid
30 fluid.gamma = 1.4;
31 fluid.R = 287;
32 fluid.gas=Solution('ndodecane_mech.cti');
33 fluid.fuel.name='nC12H26';
34 fluid.fuel.nC=12;
35 fluid.fuel.nH=26;
36
37 %define altitudes and speeds
38 altitudes = [0,9144,13106]; % [m]
39 UInf = [90 150 200]; % [m/s]
40 phi_ratio = (0.15:0.1:10); % [-]
41
42 %% Input of equivalence ratio from problem 1
43 nAltitudes = length(altitudes);
44 S_phi = zeros(1,nAltitudes);
45
46 for iAltitude = 1:nAltitudes
47     [~,~,~, sSL.r]=atmosisa(0);
48     [~,~,~, s0.r]=atmosisa(altitudes(iAltitude));
49     dmdt=HondaJet.engine.mDotSeaLevel*s0.r/sSL.r; % [kg/s]
50     [s_phi_up, s_phi_low]=matchTemperature(dmdt,UInf(iAltitude),altitudes(
           iAltitude),HondaJet.engine,fluid);
51     S_phi(1,iAltitude)=s_phi_up;
52     S_phi(2,iAltitude)=s_phi_low;
```

```
end

%% Solve problem 2b
weight=HondaJet.airframe.W/9.81;
weight_fuel= 5600/9.81;
t=0;
var_altitude=2;
[~,~,~, sSL.r]=atmosisa(0);

[~,s0.a,~, s0.r]=atmosisa(altitudes(var_altitude));
dmdt=HondaJet.engine.mDotSeaLevel*s0.r/sSL.r/4; % [kg/s]

while weight_fuel>0 % stepsize=5min
    [dmdt,mDotfuel] = matchThrust(dmdt,S_phi(2,var_altitude), altitudes(
        var_altitude), UInf(var_altitude), HondaJet, fluid,weight);
    weight_fuel=weight_fuel-mDotfuel*300;
    weight=weight-mDotfuel*300;
    t=t+5; % [min]
end

S_range=t*60*UInf(3)/1000; % [km]

U_TSFC=s0.a*0.7; % [m/s]
```

2.2 matchThrust.m

```
function [dmdt,mDotfuel_ok] = matchThrust(dmdt,phi, altitude, UInf, HondaJet,
    fluid,weight)
% Find the optimal mass flow to match
maxIt=10000;

iteration=0;
tolerance=100;
Ts_turbofan=0;
mDotfuel_ok='not assigned';

while (iteration<maxIt && tolerance>0.1 && imag(Ts_turbofan)==0)
    Ts_required=ThrustRequired(UInf,altitude,HondaJet.airframe,weight);
    [mDotfuel,~,Ts_turbofan]=realTurbofanThrust(dmdt,phi,UInf,altitude,
        HondaJet.engine,fluid);
    error_rel=(Ts_required-Ts_turbofan)/Ts_turbofan;
    if error_rel<0
        dmdt=dmdt-0.001;
    else
        dmdt=dmdt+0.0001;
    end
    mDotfuel_ok=mDotfuel;
    tolerance=abs(error_rel)*100;
    iteration=iteration+1;
end
```


3 Problem 3

3.1 Driver - Problem 3

```
% Script: problemSet3Driver
%-------------------------------------------------------------------------
%   This script is used as the driver for problem set one. Feel free to
%   tinker with the parameters to get a feel for their sensitivities, but
%   the final submission must use the parameters as is.

clear; close all; clc

%define a structure containing the HondaJet's parameters
HondaJet.airframe.S = 17.3;                          %[m^2] , reference area
HondaJet.airframe.b = 12.12;                         %[m] , wing span
HondaJet.airframe.W= 41000;                          %[N] , weight (which is max
    takeoff weight)
HondaJet.airframe.CD0 = 0.01;                        %[] , zero-lift drag
    coefficient (or parasite drag coeff.)
HondaJet.airframe.e = 0.8;                           %[] , Oswald Efficiency
HondaJet.engine.overallCompressionRatio = 24;        %[]
HondaJet.engine.turbineInletTemperature=1600;        %[K]
HondaJet.engine.mDotSeaLevel = 8.0;                  %[kg/s]
HondaJet.engine.nEngines = 2;                        %[]
HondaJet.engine.bypassRatio = 2.9;                   %[]
HondaJet.engine.fanCompressionRatio = 2.0;           %[]
HondaJet.engine.efficiencies.diffuser = 0.95;        %[]
HondaJet.engine.efficiencies.fan = 0.92;             %[]
HondaJet.engine.efficiencies.compressor = 0.87;      %[]
HondaJet.engine.efficiencies.combustor = 1.00;       %[]
HondaJet.engine.efficiencies.turbine = 0.91;         %[]
HondaJet.engine.efficiencies.coreNozzle = 0.98;      %[]
HondaJet.engine.efficiencies.fanNozzle = 0.98;       %[]

%define a structure containing the fluid
fluid.gamma = 1.4;
fluid.R = 287;
fluid.gas=Solution('ndodecane_mech.cti');
fluid.fuel.name='nC12H26';
fluid.fuel.nC=12;
fluid.fuel.nH=26;

%define altitudes and speeds
altitudes = [0,9144,13106]; % [m]
UInf = [90 150 200]; % [m/s]
phi_ratio = (0.15:0.1:5); % [-]

%% Solve problem 3a (sea level)
mDotNOmax=0.04e-3;

nphi_ratio = length(phi_ratio);
mDotNO_ratio = zeros(nphi_ratio,1);

for i=1:length(phi_ratio)
    [~,~,mDotNO_] = combustor(phi_ratio(i),UInf(1), altitudes(1), HondaJet.
        engine, fluid);
    mDotNO_ratio(i)=mDotNO_/mDotNOmax;
end
%plotNitricOxid(phi_ratio,mDotNO_ratio,altitudes(1));
```

```
%% Solve problem 3b

U4=11.8;
mDotN2=0.403;
mDotO=3.15e-7;
k=2.66;

L_dil=U4*mDotNOmax/mDotO/mDotN2/k;


%% Solve problem 3c
beta=[1 5 10];
mDotNO_ratio2=zeros(length(phi_ratio),length(beta)+1);
mDotNO_ratio2(:,1)=mDotNO_ratio;
for j=1:length(beta)
    for i=1:length(phi_ratio)
        [mDotNO_] = RQLCombustor(beta(j),phi_ratio(i),UInf(1), altitudes(1),
            HondaJet.engine, fluid);
        mDotNO_ratio2(i,j+1)=mDotNO_/mDotNOmax;
    end
end
plotNitricOxid2(phi_ratio,mDotNO_ratio2,beta);
```

3.2 Combustor.m

```
function [p4,T4,mDotNO] = combustor(phi,UInf, altitude, engine, fluid)
    % function: combustor
    % ------------------------------------------------------------------
    % This function returns the turbine inlet temperature and mass
    % fractions.
    %   Inputs:
    %       phi = equivalence ratio
    %       UInf = free stream velocity [m/s]
    %       engine = structure containing the engine parameters
    %       fluid = structure containing the fluid properties including the
    %           cantera solution object
    %   Outputs:
    %       T4 = The turbine inlet temperature [K]
    %       mDot4 = composition of the turbine inlet gas (in mass flow) [kg/s]
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %TODO:  Calculate the temperature (T4) and outlet mass flow rates from
    %       the combustor (mDot4). Note that any changes to the cantera gas
    %       object fluid.gas remain even after function calls. The
    %       combustion should occur via an HP reactor. That is use
    %           equilibrate(fluid.gas,'HP')
    %       where fluid.gas is the cantera phase object at T3,p3,X3.
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % Given states
    [s0.T, s0.a, s0.p, s0.r]=atmosisa(altitude);
    [~, ~, ~, sSL.r]=atmosisa(0);

    % Given mass flow rate:
    dmdt=engine.mDotSeaLevel*s0.r/sSL.r;


    % Compute combustion temperature
    setCombustorInletConditions(phi,UInf, altitude, engine, fluid);
    f = fuelAirRatio(fluid);
```



```

    equilibrate(fluid.gas,'HP');
    T4=temperature(fluid.gas);
    p4=pressure(fluid.gas);
    xi_NO=massFraction(fluid.gas,'no');

    % Compute mass flow rate
    mDot4=dmdt.*(1+f);
    mDotNO=xi_NO*mDot4;
end
function setCombustorInletConditions(phi,UInf, altitude, engine, fluid)
    % function: setCombustorInletConditions
    % ------------------------------------------------------------------
    % This function sets the fluid.gas solution object to the combustor
    % inlet conditions after the fuel is mixed with the oxidizer
    %   Inputs:
    %       phi = equivalence ratio
    %       UInf = free stream velocity [m/s]
    %       engine = structure containing the engine parameters
    %       fluid = structure containing the fluid properties including the
    %           cantera solution object
    %   Outputs:
    %       fluid.gas object is altered to inlet state
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %TODO:  Calculate the inlet conditions to the combustor:
    %       T3,p3
    %       The provided code sets the properties of the inlet gas. Note
    %       that the fluid object is assumed to have additional fields:
    %       fluid.gas = cantera gas object
    %       fluid.fuel.name = fuel name in cti file (e.g., 'nc12h26')
    %       fluid.fuel.nC = #number of hydrogen atoms in fuel (e.g., 12)
    %       fluid.fuel.nH = #number of hydrogen atoms in fuel (e.g., 26)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %Compute the inlet gas condition
    % Given states
    [s0.T, s0.a, s0.p, s0.r]=atmosisa(altitude);
    s0.u=UInf;
    s0.M=UInf./s0.a;

    % Given fluid
    air.R=fluid.R;
    air.g=fluid.gamma;
    air.cp=air.R* air.g/(air.g-1);

    % Given pressure ratios:
    ratio_c=engine.overallCompressionRatio;

    % Given efficiencies:
    e.diff=engine.efficiencies.diffuser;
    e.fan=engine.efficiencies.fan;
    e.cop=engine.efficiencies.compressor;

    % Solve Diffusor
    s2.T=s0.T.*( 1 +(air.g-1)/2 *s0.M.^2 );
    s2.p=s0.p.*( 1+ e.diff*(s2.T./s0.T-1)) .^(air.g/(air.g-1)) ;

    % Solve compressor
    s3.p=ratio_c.*s2.p;
```

```
    s3.T=s2.T.*(1+ 1/e.cop/e.fan *(ratio_c^((air.g-1)/air.g) -1) );

    T3=s3.T;
    p3=s3.p;


    %set the inlet gas
    nSp = nSpecies(fluid.gas);
    X3 = zeros(nSp,1);
    iFuel = speciesIndex(fluid.gas,fluid.fuel.name);
    X3(iFuel)=phi;
    iO2 = speciesIndex(fluid.gas,'o2');
    iN2 = speciesIndex(fluid.gas,'n2');
    X3(iO2)= fluid.fuel.nC+fluid.fuel.nH/4.0;
    X3(iN2)=3.76*X3(iO2);
    set(fluid.gas,'T',T3,'P',p3,'X',X3)
end

function f = fuelAirRatio(fluid)
    % function: fuelAirRatio
    % -------------------------------------------------------------
    % Determines the fuel-air ratio of the unburned gas.
    %   Inputs:
    %       fluid = structure containing the fluid properties including the
    %           cantera solution object
    %   Outputs:
    %       f = fuel-air ratio
    Y = massFractions(fluid.gas);
    iFuel = speciesIndex(fluid.gas,fluid.fuel.name);
    iO2 = speciesIndex(fluid.gas,'o2');
    iN2 = speciesIndex(fluid.gas,'n2');
    f = Y(iFuel)/(Y(iO2)+Y(iN2));
end
```

3.3 RQLCombustor.m

```
function [mDotNO] = RQLCombustor(beta,phi,UInf, altitude, engine, fluid)
    % function: RQLcombustor
    % -------------------------------------------------------------
    % This function returns the output mass flow vector as a function of
    % the rerouting ratio.
    %   Inputs:
    %       phi = the overall equivalence ratio for the combustor
    %       UInf = free stream velocity [m/s]
    %       engine = structure containing the engine parameters
    %       fluid = structure containing the fluid properties including the
    %           cantera solution object
    %   Outputs:
    %       mDot4 = composition of the turbine inlet gas (in mass flow) [kg/s]
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %TODO: Calculate the temperature (T4) and outlet mass flow rates from
    %       the combustor (mDot4). Note that any changes to the cantera gas
    %       object fluid.gas remain even after function calls. The
    %       combustion should occur in three parts:
    %       1) a call to equilibrate(fluid.gas,'HP') to simulate adiabatic
    %           combustion at T3,p3,X3 (make sure to account for the
    %           rerouted gas in X3!)
    %       2) Quenching and dilution. NO should be removed from the
    %           equilibrated gas, and the gas should be diluted with the
```



```
    %           rerouted gas at state 3 based on a weighted average using
    %           the rerouteRatio (beta_c).
    %       3) The diluted gas should be equilibrated again using
    %           equilibrate(fluid.gas,'HP'). This will be the output state
    %           of the gas.
    %       Make sure you pass in the reroute ratio. This can either be
    %       done by modifying the number of function inputs or to modify
    %       the engine object to hold this parameter.
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % massFractions(fluid.gas)
    % speciesNames(fluid.gas)

    % Given states
    [s0.T, s0.a, s0.p, s0.r]=atmosisa(altitude);
    [~, ~, ~, sSL.r]=atmosisa(0);

    % Given mass flow rate:
    dmdt=engine.mDotSeaLevel*s0.r/sSL.r;


    % 1)
    [p3,T3]=setCombustorInletConditions(phi,UInf, altitude, engine, fluid,
        beta);
    f = fuelAirRatio(fluid);
    m_a2=dmdt/(1+1/beta);
    m_a1=dmdt/(1+beta);

    m_ini=m_a1/(1-massFraction(fluid.gas,'nc12h26'));
    y_ini=massFractions(fluid.gas);
    y_air=y_ini;
        y_air(speciesIndex(fluid.gas,'nc12h26'))=0;
    setMassFractions(fluid.gas,y_air,'norm');
    cp_air=cp_mass(fluid.gas);
    setMassFractions(fluid.gas,y_ini,'norm');

    % Compute reaction
    equilibrate(fluid.gas,'HP');


    % 2)
    % delete NO-component
    y_burn= massFractions(fluid.gas);

    y_burn(speciesIndex(fluid.gas,'no'))=0;
    setMassFractions(fluid.gas,y_burn,'norm');
    cp_burn=cp_mass(fluid.gas);
    T4=temperature(fluid.gas);

    % Get mixture
    y_dil=y_burn*m_ini+y_air*m_a2;
    setMassFractions(fluid.gas,y_dil,'norm');
    cp_dil=cp_mass(fluid.gas);

    % Get temperature of mixture
    p_dil=p3;
    m_dil=m_ini+m_a2;
        % Enthalpy equilibrium
    T_dil=1/cp_dil/m_dil*(cp_air*T3*m_a2+cp_burn*T4*m_ini);
```

```
82
83      set(fluid.gas,'T',T_dil,'P',p_dil)
84
85      % 3) Compute reaction
86      equilibrate(fluid.gas,'HP');
87      xi_NO=massFraction(fluid.gas,'no');
88      mDotNO=xi_NO*m_dil;
89  end
90
91  function [p3,T3]=setCombustorInletConditions(phi,UInf, altitude, engine, fluid
        , beta)
92      % function: setCombustorInletConditions
93      % ------------------------------------------------------------
94      % This function sets the fluid.gas solution object to the combustor
95      % inlet conditions after the fuel is mixed with the oxidizer
96      %   Inputs:
97      %       phi = equivalence ratio
98      %       UInf = free stream velocity [m/s]
99      %       engine = structure containing the engine parameters
100     %       fluid = structure containing the fluid properties including the
101     %           cantera solution object
102     %   Outputs:
103     %       fluid.gas object is altered to inlet state
104     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
105     %TODO:  Calculate the inlet conditions to the combustor:
106     %       T3,p3
107     %       This function should be equivalent to that used for the
108     %       combustor. You may find it to be useful to have this as a
109     %       stand-alone function, which is shared between the two.
110     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
111
112
113         %Compute the inlet gas condition
114     % Given states
115     [s0.T, s0.a, s0.p, s0.r]=atmosisa(altitude);
116     [~, ~, ~, sSL.r]=atmosisa(0);
117     s0.u=UInf;
118     s0.M=UInf./s0.a;
119
120     % Given fluid
121     air.R=fluid.R;
122     air.g=fluid.gamma;
123     air.cp=air.R* air.g/(air.g-1);
124
125     % Given pressure ratios:
126     ratio_c=engine.overallCompressionRatio;
127
128     % Given efficiencies:
129     e.diff=engine.efficiencies.diffuser;
130     e.fan=engine.efficiencies.fan;
131     e.cop=engine.efficiencies.compressor;
132
133     % Solve Diffusor
134     s2.T=s0.T.*( 1 +(air.g-1)/2 *s0.M.^2 );
135     s2.p=s0.p.*( 1+ e.diff*(s2.T./s0.T-1)) .^(air.g/(air.g-1)) ;
136
137     % Solve compressor
138     s3.p=ratio_c.*s2.p;
139     s3.T=s2.T.*(1+ 1/e.cop/e.fan *(ratio_c^((air.g-1)/air.g) -1) );
```



```
140
141     T3=s3.T;
142     p3=s3.p;
143
144
145     %set the inlet gas
146     nSp = nSpecies(fluid.gas);
147     X3 = zeros(nSp,1);
148     iFuel = speciesIndex(fluid.gas,fluid.fuel.name);
149     X3(iFuel)=phi;
150     iO2 = speciesIndex(fluid.gas,'o2');
151     iN2 = speciesIndex(fluid.gas,'n2');
152     X3(iO2)= fluid.fuel.nC+fluid.fuel.nH/4.0*1/(1+beta);
153     X3(iN2)=3.76*X3(iO2)*1/(1+beta);
154     set(fluid.gas,'T',T3,'P',p3,'X',X3)
155 end
156
157 function f = fuelAirRatio(fluid)
158     % function: fuelAirRatio
159     % ------------------------------------------------------------
160     % Determines the fuel-air ratio of the unburned gas.
161     %   Inputs:
162     %       fluid = structure containing the fluid properties including the
163     %           cantera solution object
164     %   Outputs:
165     %       f = fuel-air ratio
166     Y = massFractions(fluid.gas);
167     iFuel = speciesIndex(fluid.gas,fluid.fuel.name);
168     iO2 = speciesIndex(fluid.gas,'o2');
169     iN2 = speciesIndex(fluid.gas,'n2');
170     f = Y(iFuel)/(Y(iO2)+Y(iN2));
171 end
```