Table of Contents

ME140 PW1000G Cycle Analysis Matlab Code 1
Introduce relevant conversion factors 1
Import Problem Data Structures 1
Problem 2 1
Problem 3 3
Problem 4 5

ME140 PW1000G Cycle Analysis Matlab Code

```
clear all; close all; clc;
```

Introduce relevant conversion factors

```
bar2pascal = 100000;
km2m = 1000;
```

Import Problem Data Structures

Parameters for Cruise and SLS Table 1

```
T1.alt = [10.67*km2m, 0*km2m];
T1.T_0 = [218.8, 288.15];
T1.P_0 = [0.239*bar2pascal, 1.014*bar2pascal];
T1.mach = [0.78, 0];
T1.fanCompRatio = [32, 28];
T1.fanRatio = [1.55, 1.52];
T1.T_o4 = [1450, 1650];
T1.mDot = [110, 265];
T1.bypassRatio = [10, 10];

% Engine Component Data
% Table 2
T2.presRecFac = [0.998, 1];
T2.fanEff = 0.95;
T2.compEff = 0.89;
T2.turbEff = 0.9;
T2.coreNozEff = 0.95;
T2.combEff = 0.95;
```

Problem 2

Introduce relevant variables

```
P2.k = 1.4;
P2.Cp = 1.005e3; %J/kgK
P2.isenExp = (P2.k-1)./P2.k; % exponent used for isentropic
 relationships
```

```
P2.R = 287.1429;%J/kgK
P2.LHV = 42.8e6; % [J/kg] fuel lower heatinv value to find Qdot for
 burner

% Inlet mach
P2.mach_0 = T1.mach;

% Mass flow through fan
P2.mDot_2 = T1.mDot; % renaming mass flow at inlet for ease of use
P2.mDot_13 = P2.mDot_2 .* T1.bypassRatio./(T1.bypassRatio + 1); % mass
 flow through bypass fan related to mas flow at inlet by bypass ratio
P2.mDot_8 = P2.mDot_2 .* 1./(T1.bypassRatio + 1);

% Pressure at inlet and beyond fan
P2.P_0 = T1.P_0;
P2.P_o0 = P2.P_0 .* presRatio(P2.k, P2.mach_0);
P2.P_o2 = P2.P_o0 .* T2.presRecFac; % Actual stagnation pressure just
 before fan
P2.P_o13 = P2.P_o2 .* T1.fanRatio; % Actual stagnation pressure after
 fan
P2.P_18 = P2.P_0;

% Temperature at inlet and beyond fan
P2.T_0 = T1.T_0;
P2.T_o0 = P2.T_0 .* tempRatio(P2.k, P2.mach_0);

P2.tempRecFac = T2.presRecFac .^ P2.isenExp ; % calculated from
 isentropic, ideal gas relationships
P2.T_o2 = P2.T_o0 .* P2.tempRecFac; % Actual stagnation temperature
 just before fan
P2.T_o13s = P2.T_o2 .* (P2.P_o13 ./ P2.P_o2) .^ P2.isenExp; %
 Isentropic stagnation temperature after fan
P2.T_o13 = P2.T_o2 + 1/T2.fanEff .* (P2.T_o13s - P2.T_o2); % Actual
 stagnation temperature after fan
P2.T_18s = P2.T_o13.*(P2.P_18 ./ P2.P_o13).^P2.isenExp;

% Velocity at outlet of bypass nozzle
P2.V_18 = (2*T2.coreNozEff*P2.Cp*(P2.T_o13-P2.T_18s)).^(1/2);
P2.V_0 = T1.mach .* sqrt(P2.k .* P2.R .* T1.T_0);
P2.F_TBypass = P2.mDot_13 .* (P2.V_18 - P2.V_0) ;

% Core calculations
% State 3
P2.P_o3 = P2.P_o2 .* T1.fanCompRatio;
P2.T_o3s = P2.T_o2 .* T1.fanCompRatio .^ P2.isenExp;
P2.T_o3 = (1/T2.compEff) .* (P2.T_o3s - P2.T_o2) + P2.T_o2;

% State 4
P2.P_o4 =  T2.combEff .* P2.P_o3;
P2.T_o4 = T1.T_o4;

% Energy Balance
P2.WFan = P2.mDot_2 .* P2.Cp .* (P2.T_o13 - P2.T_o2); % [J/s]
P2.WComp = P2.mDot_8 .* P2.Cp .* (P2.T_o3 - P2.T_o13); % [J/s]
```

```
P2.WTurb = P2.WFan + P2.WComp; % [J/s]

% State 5
P2.T_o5 = P2.T_o4 - P2.WTurb ./ (P2.Cp .* P2.mDot_8);
P2.T_o5s = P2.T_o4 + (P2.T_o5 - P2.T_o4) ./ T2.turbEff;
P2.P_o5s = P2.P_o4 .* (P2.T_o5s ./ P2.T_o4).^(1./P2.isenExp);
P2.P_o5 = P2.P_o5s;

%State 8
P2.P_8 = T1.P_0;
P2.T_8s = P2.T_o5 .* (P2.P_8 ./ P2.P_o5) .^ P2.isenExp;
P2.V_8 = (2*T2.coreNozEff*P2.Cp*(P2.T_o5-P2.T_8s)).^(1/2);
P2.F_TCore = P2.mDot_8 .* (P2.V_8 - P2.V_0); % Thrust from core

% Net Thrust is sum of thrust of core + thrust of bypass
P2.Qdot = P2.mDot_8 .* P2.Cp .* (P2.T_o4 - P2.T_o3); % [J/s] energy
 added from fuel in the combuster
P2.F_TNet = P2.F_TCore + P2.F_TBypass; % Thrust, F, the net forward
 force provided by the engine in N
P2.F_ST = (P2.F_TNet ./ (1e3)) ./ T1.mDot; % [kN/kg/s] Specific
 Thrust, ST, Thrust per unit of air mass flow through engine
P2.mDot_fuel = P2.Qdot ./ P2.LHV; % [kg/s] fuel mass flow rate
P2.TSFC = P2.mDot_fuel ./ P2.F_TNet; % [kg/s/kN] thrust specific fuel
 consumption
```

Problem 3

Introduce relevant variables

```
P3.R = 287; %J/kgK
P3.LHV = 42.8e6; % [J/kg] fuel lower heating value to find Qdot for
 burner

% Inlet mach
P3.mach_0 = T1.mach;

% Mass flow through fan
P3.mDot_2 = T1.mDot; % renaming mass flow at inlet for ease of use
P3.mDot_13 = P3.mDot_2 .* T1.bypassRatio./(T1.bypassRatio + 1); % mass
 flow through bypass fan related to mas flow at inlet by bypass ratio
P3.mDot_8 = P3.mDot_2 .* 1./(T1.bypassRatio + 1);

% State 0 givens
P3.P_0 = T1.P_0;
P3.T_0 = T1.T_0;
[P3.Cp_0, P3.k_0] = specHeatAir(P3.T_0);

% Pressure at inlet and beyond fan
P3.P_o0 = P3.P_0 .* presRatio(P3.k_0, P3.mach_0);
P3.P_o2 = P3.P_o0 .* T2.presRecFac; % Actual stagnation pressure just
 before fan
P3.P_o13 = P3.P_o2 .* T1.fanRatio; % Actual stagnation pressure after
 fan
```

```
P3.P_18 = P3.P_0;

% Temperature at inlet and beyond fan
P3.T_0 = T1.T_0;
P3.T_o0 = P3.T_0 .* tempRatio(P3.k_0, P3.mach_0);

% Iterate to determine stagnation temperature at state 2
% Finding P3.T_o2, knowing P3.T_o0, P3.P_o0, and P3.P_o2;
% using the iterative technique/function
P3.T_o2(1) = presRatio2Temp(P3.P_o0(1), P3.P_o2(1), P3.T_o0(1));
P3.T_o2(2) = presRatio2Temp(P3.P_o0(2), P3.P_o2(2), P3.T_o0(2));

% Iterate to determine stagnation temperature at state 2
P3.T_o13s(1) = presRatio2Temp(P3.P_o2(1), P3.P_o13(1), P3.T_o2(1));
P3.T_o13s(2) = presRatio2Temp(P3.P_o2(2), P3.P_o13(2), P3.T_o2(2));
P3.T_o13 = varTempEffComp(P3.T_o13s,P3.T_o2,T2.fanEff); % Actual
 stagnation temperature after fan

P3.T_18s(1) = presRatio2TempInit(P3.P_18(1), P3.P_o13(1),
 P3.T_o13(1));
P3.T_18s(2) = presRatio2TempInit(P3.P_18(2), P3.P_o13(2),
 P3.T_o13(2));

% Velocity at outlet of bypass nozzle
P3.V_0 = T1.mach .* sqrt(P3.k_0 .* P3.R .* T1.T_0);
% implemented integral method of finding delta h
P3.V_18 = (2*T2.coreNozEff*dh(P3.T_o13, P3.T_18s)).^(1/2);
P3.F_TBypass = P3.mDot_13 .* (P3.V_18 - P3.V_0);

% Core calculations
% State 3
P3.P_o3 = P3.P_o2 .* T1.fanCompRatio;
% Iterate to determine stagnation temperature at state 3
P3.T_o3s(1) = presRatio2Temp(P3.P_o2(1), P3.P_o3(1), P3.T_o2(1));
P3.T_o3s(2) = presRatio2Temp(P3.P_o2(2), P3.P_o3(2), P3.T_o2(2));
P3.T_o3 = varTempEffComp(P3.T_o3s, P3.T_o2, T2.compEff);

% State 4
P3.P_o4 =  T2.combEff .* P3.P_o3;
P3.T_o4 = T1.T_o4;

% Energy Balance
P3.WFan = P3.mDot_2 .* dh(P3.T_o13,P3.T_o2);
P3.WComp = P3.mDot_8 .* dh(P3.T_o3,P3.T_o13);
P3.WTurb = P3.WFan + P3.WComp;

% State 5  this needs to be fixed to fit with variable cp
% iterative LHS vs RHS method;
% looking for P3.T_o5 (final temp), based on work, P3.T_o4 (init temp)
% knowing the work output w_turb = W_turb/m_dot = delta h:

P3.T_o5(1) = workOut2Temp(P3.mDot_8(1), P3.WTurb(1), P3.T_o4(1));
P3.T_o5(2) = workOut2Temp(P3.mDot_8(2), P3.WTurb(2), P3.T_o4(2));
P3.T_o5s = varTempEffTurb(P3.T_o5, P3.T_o4, T2.turbEff);
```

```
% knowing the initial and final temp, using the integral, we can find
 the
% pressure ratio and the ideal final pressure
P3.P_o5s = TempRatio2Pres(P3.T_o4, P3.T_o5s, P3.P_o4);
P3.P_o5 = P3.P_o5s;

%State 8
P3.P_8 = T1.P_0;
P3.T_8s(1) = presRatio2TempInit(P3.P_8(1), P3.P_o5(1), P3.T_o5(1));
P3.T_8s(2) = presRatio2TempInit(P3.P_8(2), P3.P_o5(2), P3.T_o5(2));
% implemented integral method of finding delta h
P3.V_8 = (2* T2.coreNozEff * dh(P3.T_o5, P3.T_8s)).^(1/2);
% Velocity at the outlet of the core nozzle
P3.F_TCore = P3.mDot_8 .* (P3.V_8 - P3.V_0); % Thrust from core

% Net Thrust is sum of thrust of core + thrust of bypass
P3.F_TNet = P3.F_TCore + P3.F_TBypass; % Thrust, F, the net forward
 force provided by the engine in N
P3.F_ST = (P3.F_TNet ./ (1e3)) ./ T1.mDot; % [kN/kg/s] Specific
 Thrust, ST, Thrust per unit of air mass flow through engine
P3.Qdot = P3.mDot_8 .* dh(P3.T_o4,P3.T_o3);
P3.mDot_fuel = P3.Qdot ./ P3.LHV; % [kg/s] fuel mass flow rate
P3.TSFC = P3.mDot_fuel ./ P3.F_TNet; % [kg/s/kN] thrust specific fuel
 consumption
```

Problem 4

```
% Missing values from above
P3.T_8 = [findT_8(P3.V_8(1), P3.T_8s(1), P3.T_o5(1)),...
        findT_8(P3.V_8(2), P3.T_8s(2), P3.T_o5(2))];
[~, P3.k_8] = specHeatAir(P3.T_8);
P3.M_8 = P3.V_8./(sqrt(P3.k_8.*P3.R.*P3.T_8));
P3.T_o8 = P3.T_8.*(1+(P3.k_8-1)./2.*P3.M_8.^2);
P3.P_o8 = P3.P_8.*(1+(P3.k_8-1)./2.*P3.M_8.^2).^(P3.k_8./(P3.k_8 -
 1));

% Setting up to pass into function
P4.T_core = [P3.T_0; P3.T_o0; P3.T_o2; P3.T_o3; P3.T_o4; P3.T_o5; ...
             P3.T_o8; P3.T_8];
P4.P_core = [P3.P_0; P3.P_o0; P3.P_o2; P3.P_o3; P3.P_o4; P3.P_o5; ...
             P3.P_o8; P3.P_8];
% Using function to find entropy points
P4.s_core = entropy(P4.T_core, P4.P_core);

figure('units','normalized','outerposition',[0 0 .75 .75]); % for
 larger plot
plot(P4.s_core(:,1),
 P4.T_core(:,1),'bo--','LineWidth',2,'MarkerSize',5)
title('T-s Diagram Core Flow at Design Cruise Conditions');
xlabel('Specific Entropy, s [J/
kgK]','FontSize',18,'FontWeight','bold');
ylabel('Temperature, T [K]','FontSize',18,'FontWeight','bold');
```

```
set(gca,'FontSize',18)
a = ['0 '; 'o0'; 'o2'; 'o3'; 'o4'; 'o5'; 'o8'; '8 ']; c = cellstr(a);
dx = [25; 25; 25; 25; 25; -75 ; 25 ; 25];
dy = [25; -50; 50; -25; -25; -25 ; -25 ; -25];
% displacement so the text does not overlay the data points
text(P4.s_core(:,1)+ dx, P4.T_core(:,1)+dy, c, 'FontSize',18);

figure('units','normalized','outerposition',[0 0 .75 .75]); % for
 larger plot
plot(P4.s_core(:,2),
 P4.T_core(:,2),'bo--','LineWidth',2,'MarkerSize',5)
title('T-s Diagram of Core Flow at SLS Conditions');
xlabel('Specific Entropy, s [J/
kgK]','FontSize',18,'FontWeight','bold');
ylabel('Temperature, T [K]','FontSize',18,'FontWeight','bold');
set(gca,'FontSize',18)

dx = [25; 60; 95; 25; 25; -75 ; 25 ; 25];
dy = [25; 25; 25; 0; -25; -25 ; -25 ; -25];
% displacement so the text does not overlay the data points
text(P4.s_core(:,2)+dx, P4.T_core(:,2)-dy, c, 'FontSize',18);

% Missing values from above;
P3.P_18 = T1.P_0;

P3.T_18 = [findT_8(P3.V_18(1), P3.T_18s(1), P3.T_o13(1)),...
          findT_8(P3.V_18(2), P3.T_18s(2), P3.T_o13(2))];

[~, P3.k_18] = specHeatAir(P3.T_18);
P3.M_18 = P3.V_18./(sqrt(P3.k_18.*P3.R.*P3.T_18));

P3.T_o18 = P3.T_18.*(1+(P3.k_18-1)./2.*P3.M_18.^2);
P3.P_o18 = P3.P_18.*(1+(P3.k_18-1)./2.*P3.M_18.^2).^(P3.k_18./(P3.k_18
 - 1));

% Setting up to pass into function
P4.T_bypass = [P3.T_0; P3.T_o0; P3.T_o2; P3.T_o13; P3.T_o18; P3.T_18];
P4.P_bypass = [P3.P_0; P3.P_o0; P3.P_o2; P3.P_o13; P3.P_o18; P3.P_18];
% Using function to find entropy points
P4.s_bypass = entropy(P4.T_bypass, P4.P_bypass);

figure('units','normalized','outerposition',[0 0 .75 .75]); % for
 larger plot
plot(P4.s_bypass(:,1),
 P4.T_bypass(:,1),'bo--','LineWidth',2,'MarkerSize',5)
title('T-s Diagram of Bypass Flow at Design Cruise Conditions');
xlabel('Specific Entropy, s [J/
kgK]','FontSize',18,'FontWeight','bold');
ylabel('Temperature, T [K]','FontSize',18,'FontWeight','bold');
set(gca,'FontSize',18)

a = ['0  '; 'o0 '; 'o2 '; 'o13'; 'o18'; '18 ']; c = cellstr(a);
dx = [0.2; 0; .2; 0; -0.5; 0];
dy = [2; 5; -5; 4; 2; -5];
```

```
% displacement so the text does not overlay the data points
text(P4.s_bypass(:,1)+ dx, P4.T_bypass(:,1)+ dy, c,'FontSize',18);

figure('units','normalized','outerposition',[0 0 .75 .75]); % for
 larger plot
plot(P4.s_bypass(:,2),
 P4.T_bypass(:,2),'bo--','LineWidth',2,'MarkerSize',5)
title('T-s Diagram of Bypass Flow at SLS Conditions');
xlabel('Specific Entropy, s [J/
kgK]','FontSize',18,'FontWeight','bold');
ylabel('Temperature, T [K]','FontSize',18,'FontWeight','bold');
set(gca,'FontSize',18)

dx = [0.2; 0.7; 1.2; 0; 0; 0.5];
dy = [0; 0; 0; 2; 2; 2];
% displacement so the text does not overlay the data points
text(P4.s_bypass(:,2)+dx, P4.T_bypass(:,2)+dy, c,'FontSize',18);
```

Published with MATLAB® R2017b