Table of Contents

Project 2 Analysis

This script is the main controller for the SR30 analysis in Project 2 of ME140. It should import data, and call the component functions for each of the individual parts of the project.

Authors: Jean-Christophe Perrin, Beck Goodloe, Richard Randall, Jason Trinidad

Created: 2018-04-11 Edited: 2018-04-16

```
clear all;
clc;
```

Useful Constants

These are mostly conversion factors so that we can convert from imperial units collected into metric.

```
const.insqToMsq = 0.00064516; % [m^2/in^2]
const.lbfToN = 4.44822; % [N/lbf]
const.R = 287; % [kJ/kg/k] Gas constant of air
const.KCdiff = 273; % [deg]
const.kPaToPa = 1e3; % [Pa/kPa]
const.kJkmol2Jkg = 1e3/28.97; %[J/kg * kmol/kJ]
const.airCoefs = const.kJkmol2Jkg .* ...
    fliplr([28.11, 0.1967e-2, 0.4802e-5,-1.966e-9]); % [J/kg/k] cp air
const.intAirCoefs = polyint(const.airCoefs);
const.LHVJetA = 42800; % [kJ/kg]
const.kg2g = 1e3; % [kg/g]
```

Given Data

The following values were supplied to us in the original project specifications and should not be changed.

```
given.pitotEffectiveArea = 6.4*const.insqToMsq; % [m^2]
given.jetAHeating = 42.8e6; % [J/kg/K]
given.A = [27.3, 6.4, 9.0, 7.2, 4.7, 3.87]'.*const.insqToMsq; % [m^2]
```

Import Collected Data

We import the collected data into an table for use throughout our function. This table should never be changed, compromising the validity of our collected data. Instead, all deried values should be copied out the table into a seperate vector or table.

```
fname = 'data.txt';
collectedData = readtable(fname);
collectedData.Properties.VariableUnits =
 {'', 'C', 'C', 'C', 'C', 'C', ...
    'C', 'kPa', 'kPa', 'kPa', 'kPa', 'kPa', 'kg/s', 'lbs'};

nObservations = height(collectedData); % Number of observations
clear fname
```

```
Warning: Variable names were modified to make them valid MATLAB
 identifiers. The
original names are saved in the VariableDescriptions property.
```

We also measured the following values seperately.

```
T1 = 23; % [C] room temp
P1 = 101.4e3; % [Pa] atmospheric pressue
```

Solve for Each RPM Observation

Loop over every observation taken. For each set of Tm or Pm (the measured values of Temp and Pressure) call Richard's Analysis to calculated the values of interest (T0, P0, M, V). These are stored in a 2D array. Each row of the array corresponds to an observation (i.e. one RPM). Each column is a location of interest.

```
nLocations = 6; % locations of interest 1-5, 8
T0 = NaN(nObservations, nLocations);
P0 = NaN(nObservations, nLocations);
Ma = NaN(nObservations, nLocations);
V = NaN(nObservations, nLocations);

mdotAir = NaN(nObservations, 1); % [kg/s]

for iObservation = 1:nObservations
    Tm = collectedData{iObservation, 2:6};
    Pm = collectedData{iObservation, 8:12};
    mdotFuel = collectedData{iObservation, 13};
    thrust = collectedData{iObservation, 14};
    [calculatedValues, thisMdotAir] = solveEachLocation(Tm, Pm,
 mdotFuel, thrust);
    calculatedValues([6, 7], :) = [];
    mdotAir(iObservation) = thisMdotAir;
    T0(iObservation, :) = cell2mat(calculatedValues.T0)';
    P0(iObservation, :) = cell2mat(calculatedValues.P0)';
    Ma(iObservation, :) = cell2mat(calculatedValues.M)';
    V(iObservation, :) = cell2mat(calculatedValues.V)';
end

clear Tm Pm mdotFuel thrust thisMdotAir iObservation calculatedValues
```

Deliverable 2

Use your performance data and area measurements (listed in Appendix) to construct a series of plots showing how the following quantities vary with spool speed:

- station stagnation temperature (K),
- station stagnation pressure (kPa, absolute),
- station Mach number
- station velocity (m/s).

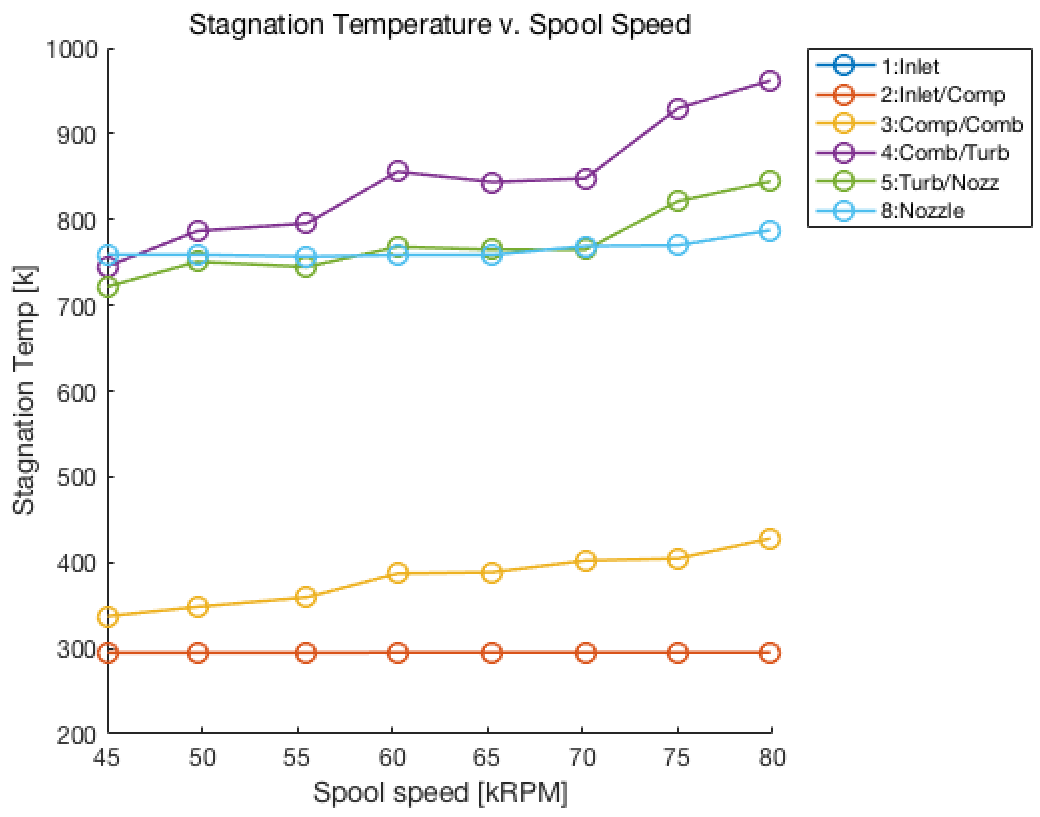
Make sure to include Station 1 in all of your plots.

First will calculate the deisred values, then plot them later on.

```
krpm = collectedData.RPM/1000; % rescale RPM for plotting

mdotFuel = collectedData.FuelFlow; % [kg/s]
airFuelRatio = mdotAir./mdotFuel; % [1]
```

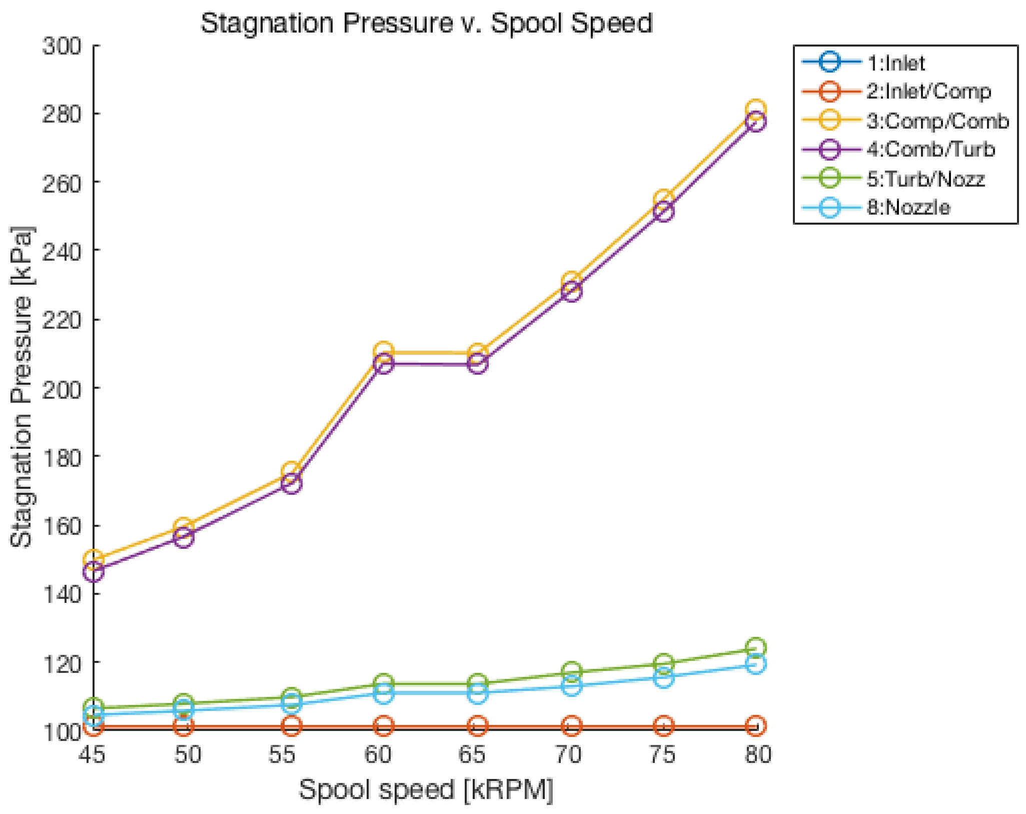
Stagnation Temp vs Spool Speed

```
legendString = {'1:Inlet', '2:Inlet/Comp', '3:Comp/Comb', '4:Comb/
Turb', ...
    '5:Turb/Nozz', '8:Nozzle'};

plot(krpm, T0, '-o');
xlabel('Spool speed [kRPM]');
ylabel('Stagnation Temp [k]');
legend(legendString, 'Location', 'bestoutside');
title('Stagnation Temperature v. Spool Speed');
plotFixer();
print('-depsc','-tiff','-r300','plots/stagTVsRpm');
```



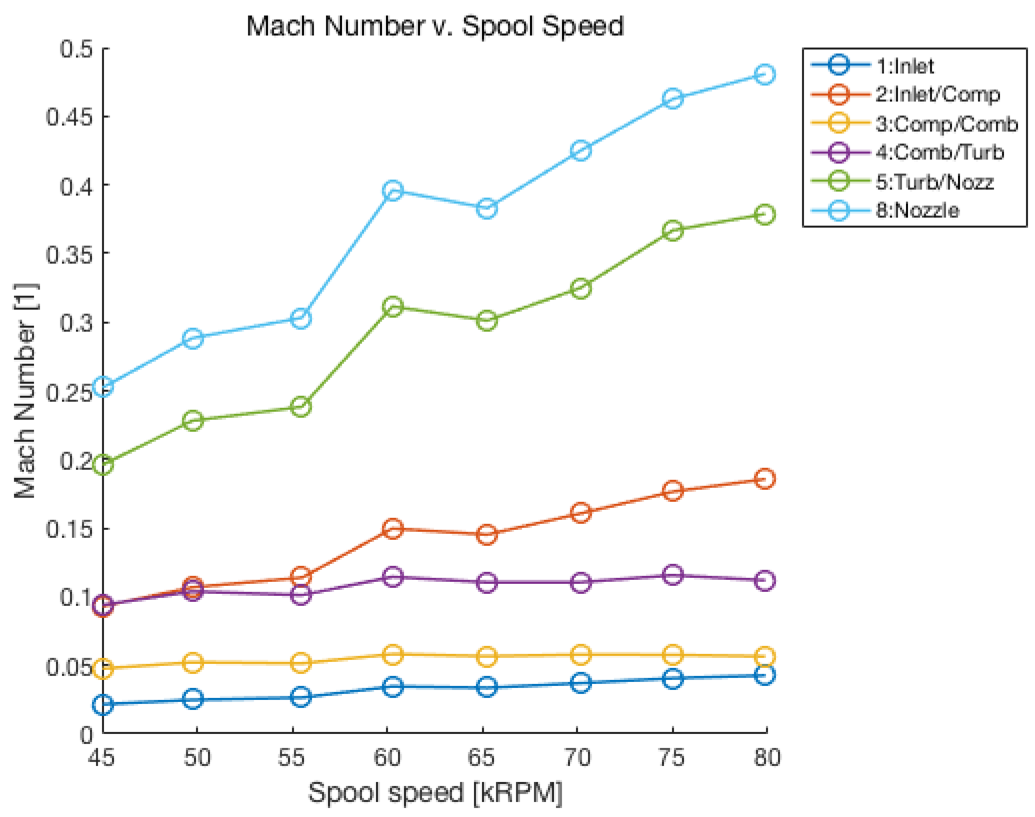
Stagnation Pressure vs Spool Speed

```
plot(krpm, 1e-3.*P0, '-o');
xlabel('Spool speed [kRPM]');
ylabel('Stagnation Pressure [kPa]');
legend(legendString, 'Location', 'bestoutside');
title('Stagnation Pressure v. Spool Speed');
plotFixer();
print('-depsc','-tiff','-r300','plots/stagPVsRpm');
```



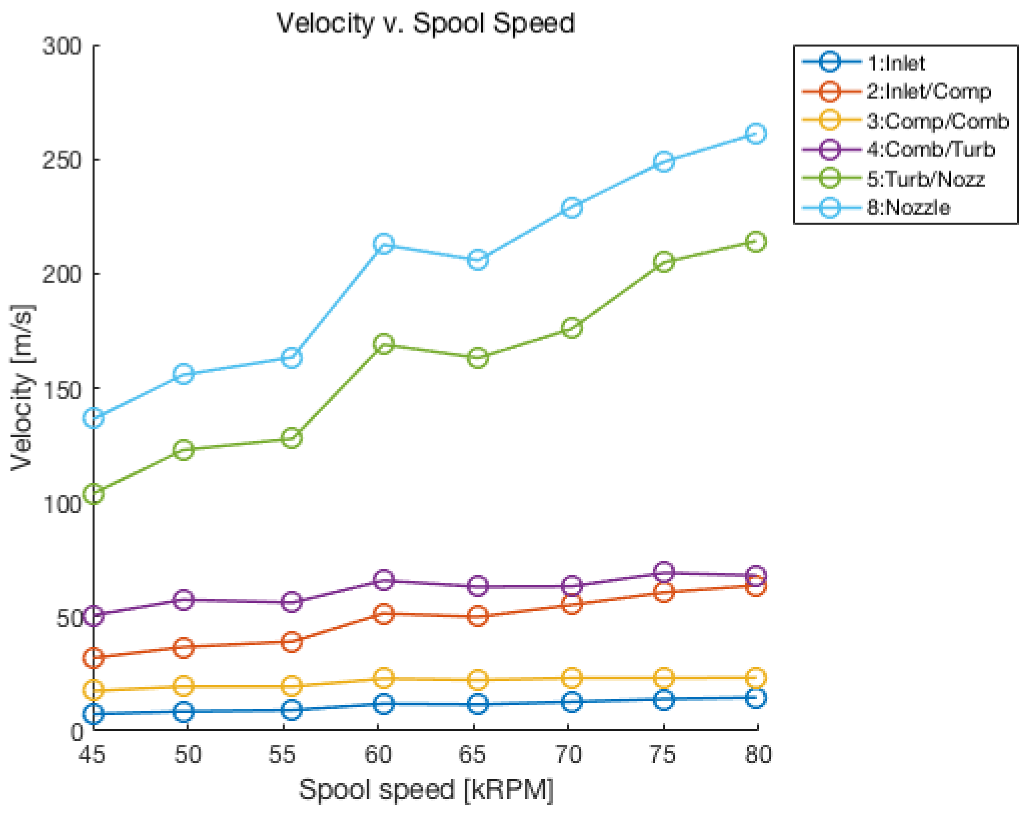
Mach Number vs Spool Speed

```
plot(krpm, Ma, '-o');
xlabel('Spool speed [kRPM]');
ylabel('Mach Number [1]');
legend(legendString, 'Location', 'bestoutside');
title('Mach Number v. Spool Speed');
plotFixer();
print('-depsc','-tiff','-r300','plots/MaVsRpm');
```



Velocity vs Spool Speed

```
plot(krpm, V, '-o');
xlabel('Spool speed [kRPM]');
ylabel('Velocity [m/s]');
legend(legendString, 'Location', 'bestoutside');
title('Velocity v. Spool Speed');
plotFixer();
print('-depsc','-tiff','-r300','plots/velVsRpm');
```



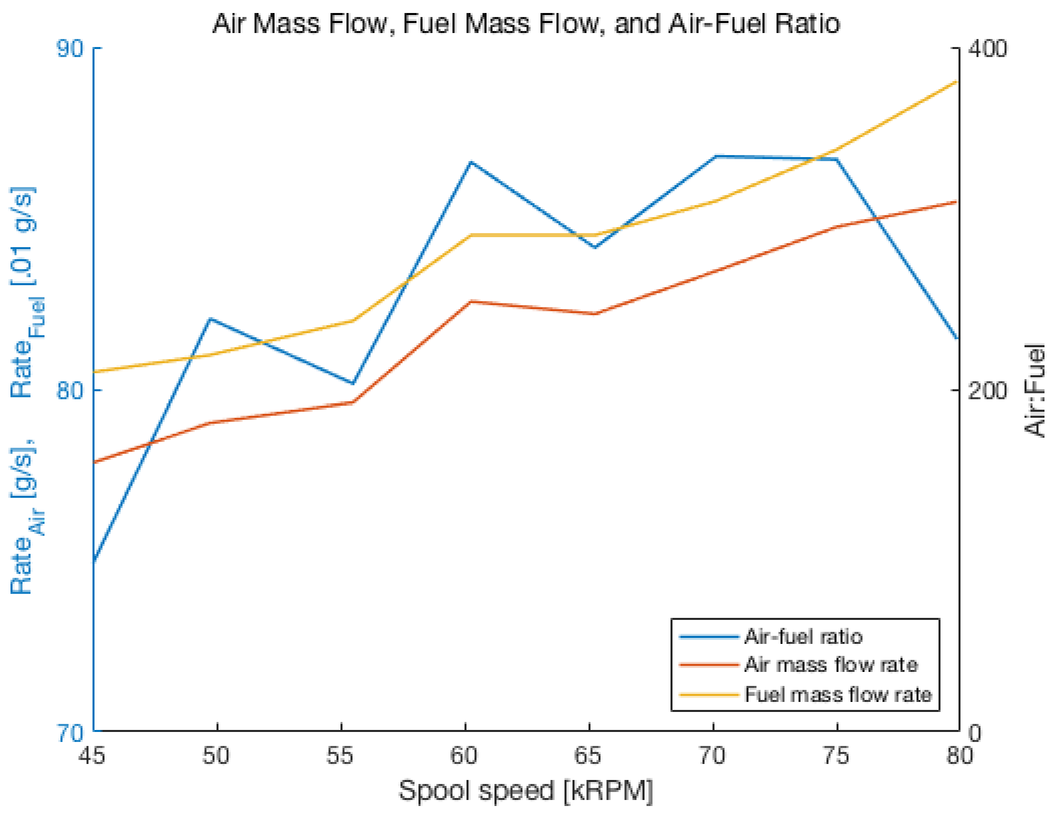
Air mass flow rate, fuel mass flow rate, air-fuel ratio

All three of these quantities to be plotted on the same plot. TA (Isabel) recommends plotting mdot_air/10 and mdot_fuel*10 so that the plot is meaningful, and indicating the change in the legend

TODO: Convert to plotyy

```
scaledMdotAir = mdotAir.*const.kg2g; % [g/s]
scaledMdotFuel = mdotFuel.*const.kg2g * 100; % 100 [g/s]

close;
figure;
hAx = plotyy(krpm, airFuelRatio, [krpm, krpm],...
                                 [scaledMdotAir, scaledMdotFuel]);

xlabel('Spool speed [kRPM]');
ylabel(hAx(1), 'Rate_{Air} [g/s],     Rate_{Fuel} [.01 g/s]') % left
 y-axis
ylabel(hAx(2), 'Air:Fuel') % right y-axis

legendString = {'Air-fuel ratio', 'Air mass flow rate', ...
    'Fuel mass flow rate'};
legend(legendString, 'Location', 'southeast');
title('Air Mass Flow, Fuel Mass Flow, and Air-Fuel Ratio');
```

```
plotFixer();
print('-depsc','-tiff','-r300','plots/mdot,ratioVsRpm');

clear hAx
```



Deliverable 3

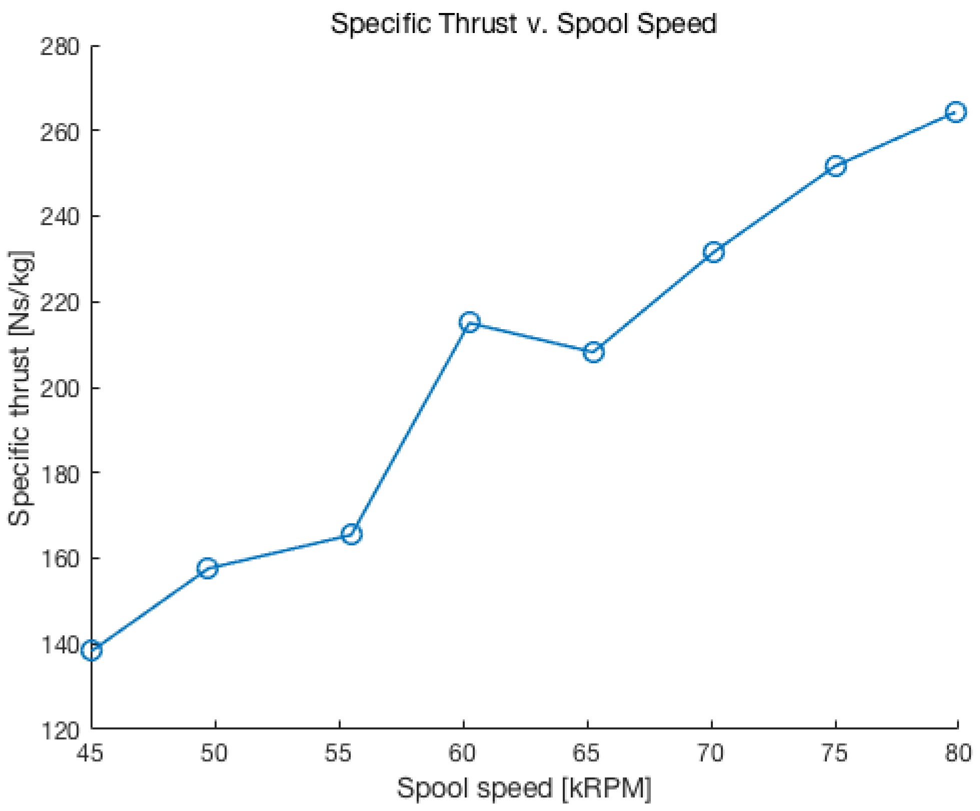
Construct performance-metric plots showing how specific thrust, thrust specific fuel consumption, and thermal efficiency vary with spool speed. Use the lower heating value of Jet A (42,800 kJ/kg) for your thermal efficiency calculation, and base your performance metrics on the calculated thrust.

```
idealThrust = (mdotAir+mdotFuel).*V(:, end);

sp_thrust = idealThrust ./ mdotAir;
TSFC = idealThrust ./ mdotFuel; % Thrust Specific Fuel Consumption

powerIn = mdotFuel*const.LHVJetA; % [kW]
powerThrust = idealThrust.*V(:,end);
thermal_eff = powerThrust ./ powerIn;

clear powerThrust powerIn
```

Specific thrust vs spool speed

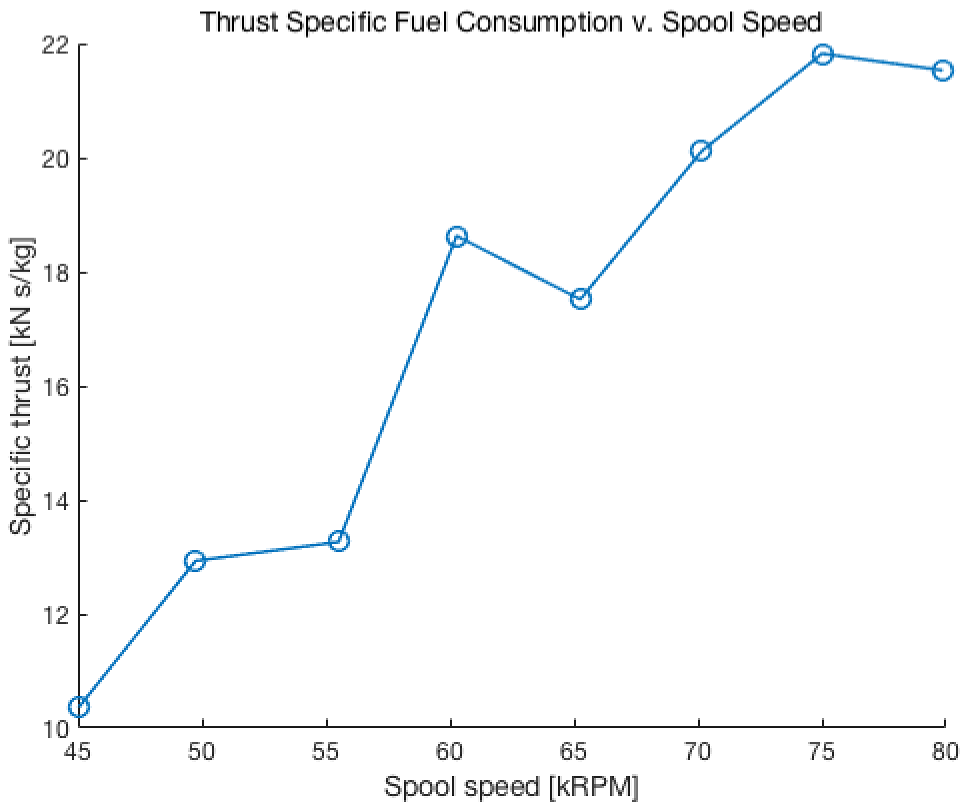
```
plot(krpm, sp_thrust, '-o');
```

```
xlabel('Spool speed [kRPM]');
ylabel('Specific thrust [Ns/kg]');
title('Specific Thrust v. Spool Speed');
plotFixer();
print('-depsc','-tiff','-r300','plots/spthrustVsRpm');
```



Thrust specific fuel consumption

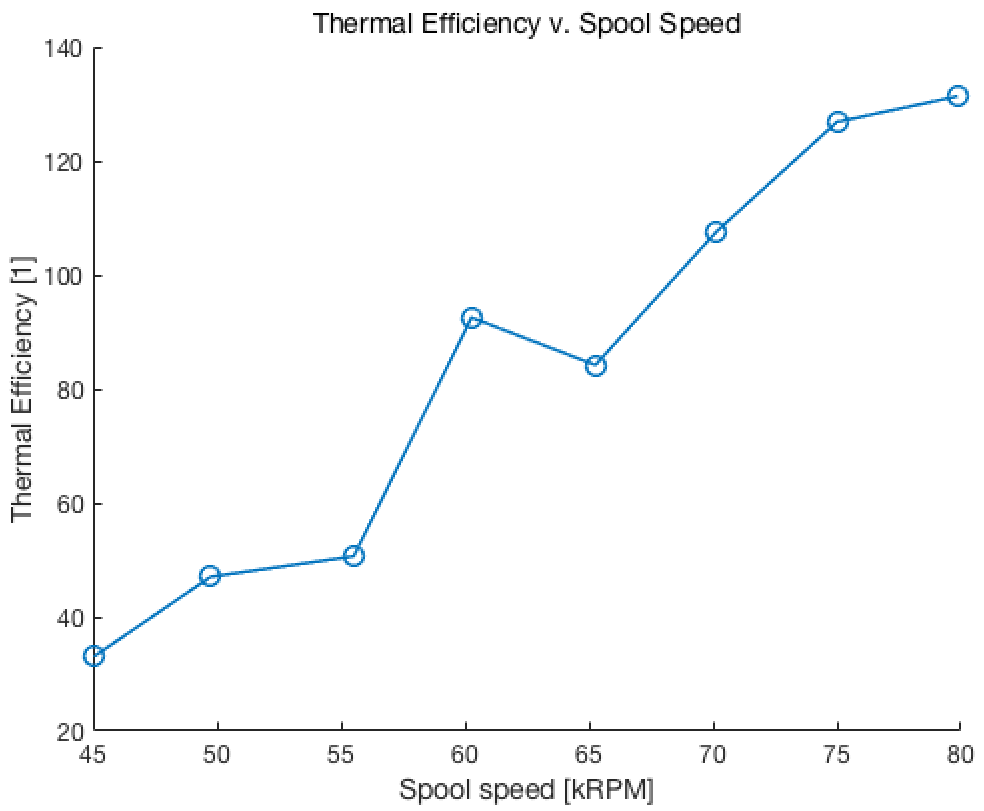
```
plot(krpm, 1e-3.*TSFC, '-o');
xlabel('Spool speed [kRPM]');
ylabel('Specific thrust [kN s/kg]');
title('Thrust Specific Fuel Consumption v. Spool Speed');
plotFixer();
print('-depsc','-tiff','-r300','plots/tsfcVsRpm');
```



Thermal Efficiency

The thermal efficiency of the engine is defined as the ratio between the useful power of the engine over the power supplied.

```
plot(krpm, thermal_eff, '-o');
xlabel('Spool speed [kRPM]');
ylabel('Thermal Efficiency [1]');
title('Thermal Efficiency v. Spool Speed');
plotFixer();
print('-depsc','-tiff','-r300','plots/thermalEffVsRpm');
```



Deliverable 4

Using your stagnation temperature data, find the power consumed by the compressor and produced by the turbine. Use these data to find the adiabatic efficiencies of these components.

Since we are modeling combustion as though it were heat transfer to air, it is also possible to define an apparent combustion efficiency as the change in enthalpy of the air (as determined from T measurements) per unit heating value of the fuel used.

Generate the following plots (vs. engine speed): * Power consumed by the compressor, Power generated by turbine * Adiabatic Efficiency of: compressor, turbine, nozzle * Stagnation pressure ratio across the combustor * Apparent Combustion Efficiency

The work flowing into a turbine should be equal to the difference in enthalpy across the turbine. Power is the rate of work so:

$$P = \dot{W} = \dot{m}\Delta \dot{h_0} = \dot{m}cp_{avg}\Delta T_0$$

We'll begin by defining some anonymous functions for the average values of specific heats.

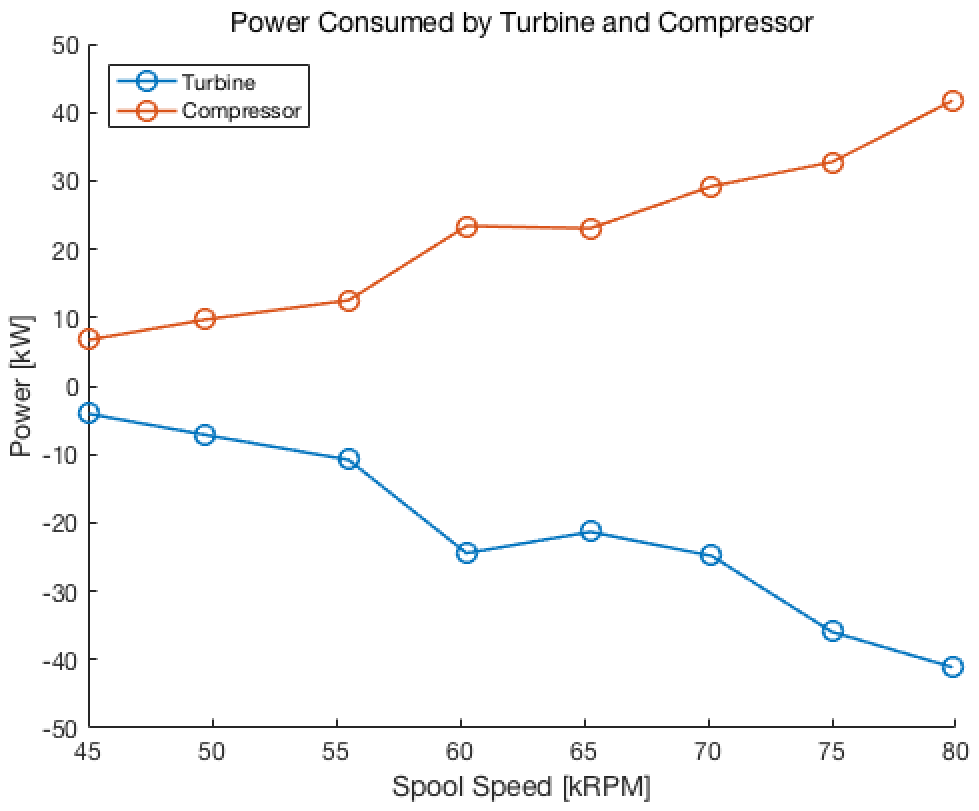
```
cp = @(T) polyval(const.airCoefs, T);
integralCp = @(T) polyval(const.intAirCoefs, T);
integrateCp = @(T1, T2) integralCp(T2)-integralCp(T1);
cp_avg = @(T1, T2) integrateCp(T1, T2) ./ (T2-T1);
gamma_avg = @(T1, T2) cp_avg(T1, T2) ./ (cp_avg(T1, T2) - const.R);
```

Now we can calculate the across the compressor and turbine based on the change in stagnation temperature across both.

```
T0TurbIn = T0(:, 4);
T0TurbOut = T0(:, 5);
dT0Turbine = T0TurbOut - T0TurbIn;

T0CompIn = T0(:, 2);
T0CompOut = T0(:, 3);
dT0Compressor = T0CompOut - T0CompIn;

PTurb = cp_avg(T0TurbIn, T0TurbOut) .* dT0Turbine .* (mdotAir +
 mdotFuel);
PComp = cp_avg(T0CompIn, T0CompOut) .* dT0Compressor .* mdotAir;
```

Then we plot both together.

```
plot(krpm, [PTurb, PComp].*1e-3, '-o');
xlabel('Spool Speed [kRPM]');
ylabel('Power [kW]');
title('Power Consumed by Turbine and Compressor');
legend('Turbine', 'Compressor', 'Location', 'northwest');
plotFixer();
print('-depsc','-tiff','-r300','plots/pTurbinepCompVsRpm');
```



Efficiencies

The adiabatic efficiencies of the compressor and turbine are given by

$$\eta = \frac{W}{W_s} = \frac{\Delta T_{0s}}{\Delta T_0}$$

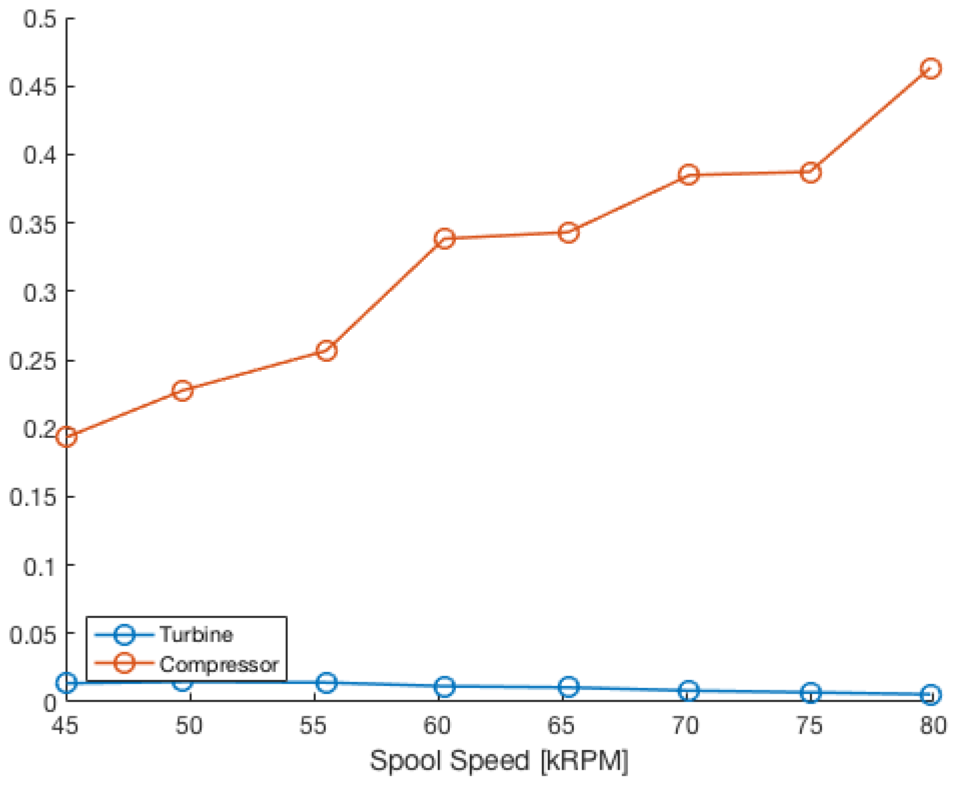
We can find the isentropic stagnation Temp using the isentropic relation

$$\frac{T_{02s}}{T_{01}} = \frac{P_{02}}{P_{01}}^{\frac{\gamma}{\gamma-1}}$$

```
k = gamma_avg(T0TurbIn, T0TurbOut);
P0In = P0(:, 4);
P0Out = P0(:, 5);
T0TurbS = T0TurbIn .* (P0In ./ P0Out).^(k./(k-1));
effTurbine = abs(dT0Turbine./(T0TurbS - T0TurbIn));

k = gamma_avg(T0CompIn, T0CompOut);
P0In = P0(:, 2);
P0Out = P0(:, 3);
T0CompS = T0CompIn .* (P0In ./ P0Out).^(k./(k-1));
effCompressor = abs(dT0Compressor./(T0CompS - T0CompIn));

clear T0CompIn T0CompOut T0TurbIn T0TurbOut dT0Turbine dT0Compressor k
clear T0CompS T0TurbS P0In P0Out
```

Plot efficiency of both components against rpm.

```
plot(krpm, [effTurbine, effCompressor], '-o');
xlabel('Spool Speed [kRPM]');
legend('Turbine', 'Compressor', 'Location', 'southwest');
plotFixer();
```



Published with MATLAB® R2017b