MATLAB Template for a Single Non-Linear Equation

By Jordan Holman

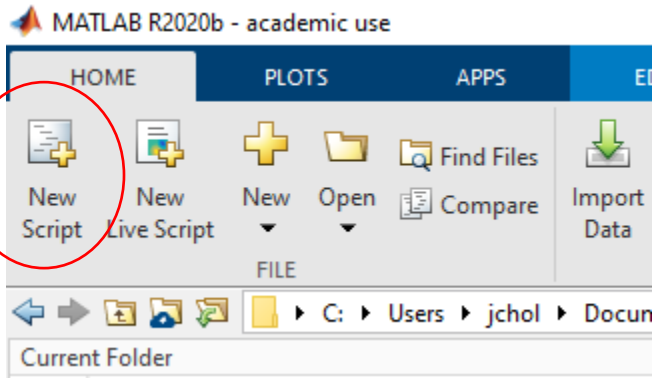
The use of this MATLAB template for solving a steady state, adiabatic combustion reaction problem will be demonstrated using an example problem similar to Example 9.6-2 from the book of Felder, Rousseau and Bullard: "Elementary Principles of Chemical Processes". The problem will be modified to gain insightful knowledge on how to use MATLAB to solve a single non-linear equation (NLE). This will require Citrix to launch the MATLAB software, unless it has already been downloaded on your computer. Below are the steps in opening MATLAB in Citrix and how to solve this problem.

Steps to Run MATLAB through Citrix:

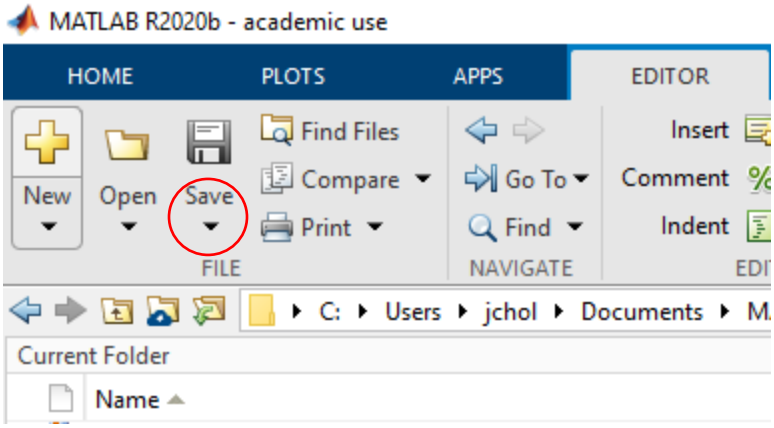
1. Open and log into Citrix.
2. Once the Citrix remote desktop is loaded, type MATLAB into the search bar in the lower left corner of the screen. Select MatlabR2020b. This will launch the MATLAB software.

Now that you have MATLAB open, you will want to make 2 new scripts. Follow the steps below to do so.

1. In the MATLAB window, locate the new script button in the upper left corner and select twice. You will need two separate scripts to solve this problem using MATLAB. Make sure you are in the HOME tab to find this button.



2. Once the two scripts are created you will want to select the EDITOR tab which is located 3 tabs to the right of HOME, then select the down arrow on SAVE and click Save As.



3. The names for these two scripts will be as follows. One script needs to be named "AdiabaticFlame" and the other needs to be named "AdiabaticFlameResult". Both of these names are **case sensitive** and misspelling them will result in an error in the following code. Additionally, it is mandatory that these two scripts be saved in the same folder, otherwise it will result in errors in the following code.

Adiabatic Flame Combustion

The objective of this problem is to obtain an outlet temperature for a steady-state, adiabatic reactor that involves a full combustion reaction. This involves components of iso-butanol, air (oxygen and nitrogen), water and carbon dioxide. Below is the reaction happening inside the reactor.

$$C_4H_9OH + \frac{13}{2}O_2 \rightarrow 4CO_2 + 5H_2O$$

Additionally, this problem requires 20% excess air to ensure that the reaction is a full combustion.

From the lecture videos, an adiabatic system means that the system does not give or lose energy, meaning $\dot{Q} = 0$ In deriving the mass and energy balance, the equation needed below is what should be obtained. Since a reaction is involved, the energy balance will need to be on a molar basis.

$$F_{in} * \hat{H}_{in} = F_{out} * \hat{H}_{out} + \cancel{\dot{Q}}$$ Adiabatic

Where F is mol/s and $\hat{H}$ is kJ/mol

The format for the rest of this problem-solving exercise will be screenshots of the code, followed by a table that explains what each line does in the code. The left side of the tables will be the code, and the right side will be an explanation to what the code does. The right side of the table is not a part of the code, as it is just an explanation.

“AdiabaticFlame” Script

Editor - C:\Users\jchol\Documents\MATLAB\AdiabaticFlame.m

AdiabaticFlame.m | AdiabaticFlameResult.m

```
1  function fT = AdiabaticFlame(T)
2  O2Excess = 1.2 * 13 / 2; %mol/s
3  NewN2 = 0.79 / 0.21 * O2Excess; %mol/s
4  Flo = 1 / 1000; %kmol/s
5  F2o = O2Excess / 1000; %kmol/s
6  F3o = NewN2 / 1000; %H2O #kmol/s
7  F4o = 0; %CO2 #kmol/s
8  F5o = 0; %kmol/s
9  %Fl = Flo - (1 / 1000); %kmol/s
10 F2 = F2o - (13 / 2 / 1000); %kmol/s
11 F3 = F3o - 0; %kmol/s
12 F4 = F4o + 5 / 1000; %kmol/s
13 F5 = F5o + 4 / 1000; %kmol/s
14 HlformLq = -334700; %kJ/kmol
15 HlformVap = -283200; %kJ/kmol
16 H2formVap = 0; %kJ/kmol
17 H3formVap = 0; %J/kmol
18 H4formVap = -241820; %kJ/kmol
19 H5formVap = -393510; %kJ/kmol
20 %Cpi=[A,B,C,D,E]
21 Cpv1=[87940,241600,1718,165400,798.7];
22 Cpv2=[29103 10040 2527 9356 1154];
23 Cpv3=[29105 8615 1702 103 910];
24 Cpv4=[33363 26790 2611 8896 1169];
25 Cpv5=[29370 34540 1428 26400 588];
26 To=25+273.15;
27 Cp2=1/1e3*(Cpv2(1)*(T-To)+Cpv2(2)*Cpv2(3)*(1/tanh(Cpv2(3)/T)-1/tanh(Cpv2(3)/To))-Cpv2(4)*Cpv2(5)*(tanh(Cpv2(5)/T)-tanh(Cpv2(5)/To)));
28 Cp3=1/1e3*(Cpv3(1)*(T-To)+Cpv3(2)*Cpv3(3)*(1/tanh(Cpv3(3)/T)-1/tanh(Cpv3(3)/To))-Cpv3(4)*Cpv3(5)*(tanh(Cpv3(5)/T)-tanh(Cpv3(5)/To)));
29 Cp4=1/1e3*(Cpv4(1)*(T-To)+Cpv4(2)*Cpv4(3)*(1/tanh(Cpv4(3)/T)-1/tanh(Cpv4(3)/To))-Cpv4(4)*Cpv4(5)*(tanh(Cpv4(5)/T)-tanh(Cpv4(5)/To)));
30 Cp5=1/1e3*(Cpv5(1)*(T-To)+Cpv5(2)*Cpv5(3)*(1/tanh(Cpv5(3)/T)-1/tanh(Cpv5(3)/To))-Cpv5(4)*Cpv5(5)*(tanh(Cpv5(5)/T)-tanh(Cpv5(5)/To)));
31 HlLiq = HlformLq;
32 H2Vap = H2formVap + Cp2;
33 H3Vap = H3formVap + Cp3;
34 H4Vap = H4formVap + Cp4;
35 H5Vap = H5formVap + Cp5;
36 Hdotin = HlLiq * Flo;
37 Hdotout = H2Vap * F2 + H3Vap * F3 + H5Vap * F5 + H4Vap * F4;
38 fT = Hdotout - Hdotin;
39 end
```

Calculated excess O_2 and N_2 needed for this problem

Inlet molar flowrates

Outlet molar flowrates with changes from reaction table

Outlet molar flowrates with changes from reaction table

Cpv coefficients obtained from DIPPR. Each is listed in an array as (A,B,C,D,E)

Inlet Temperature

Total Molar Enthalpy Values

Enthalpy inlet and outlet flows from energy balance

Shown above is the entire “AdiabaticFlame” script. Below this will be the code, along side the explanation of it. In MATLAB semicolons are recommended since they suppress the printing of the function (aka you will see the variable and what it is equal to print out in the command line). In MATLAB it is the new line that actually ends the line. Notice how each line uses a semicolon except for lines 1 and 39 because these two lines begin and end the function. In MATLAB, comments are used by typing a “%” followed by what note you want to write down. Comments are notes that help the user understand a line of code and do not impact the code itself in any way. In this script, the comments represent the units of the values that are declared. You can tell if text is commented because it will be green. This script is used to define a function. In the next script we write, we will be calling the function there. When we call the function in the next script, all the code you see here will be ran, but only takes up one line of code.

`function fT = AdiabaticFlame(T)`	This defines the non-linear block function, which will hold the equations and constants that will be needed to solve the non-linear function. fT will be the value that is returned once the function is ran, and T is called an input parameter. An input parameter is inputted by the user and will be explained later.

`O2Excess = 1.2 * 13 / 2; %mol/s`
`NewN2 = 0.79 / 0.21 * O2Excess; %mol/s`
`F1o = 1 / 1000; %kmol/s`
`F2o = O2Excess / 1000; %kmol/s`
`F3o = NewN2 / 1000; %H2O #kmol/s`
`F4o = 0; %CO2 #kmol/s`
`F5o = 0; %kmol/s`
`%F1 = F1o - (1 / 1000); %kmol/s`
`F2 = F2o - (13 / 2 / 1000); %kmol/s`
`F3 = F3o - 0; %kmol/s`
`F4 = F4o + 5 / 1000; %kmol/s`
`F5 = F5o + 4 / 1000; %kmol/s`
`H1formLq = -334700; %kJ/kmol`
`H1formVap = -283200; %kJ/kmol`
`H2formVap = 0; %kJ/kmol`
`H3formVap = 0; %J/kmol`
`H4formVap = -241820; %kJ/kmol`
`H5formVap = -393510; %kJ/kmol`
`To=25+273.15;`
	Here we are assigning variables values that we will be using in the problem. An "=" is the assignment operator in MATLAB and the variable name is always on the left, followed by its value on the right. Take this line of code for example.
	
	`F1o = 1 / 1000; %kmol/s`
	
	In this line, F1o is the variable name, and its being assigned the value of 1/1000. The semicolon ends the line, and the green text following the equation is a comment that shows the units of F1o as kmol/s.

`%Cpi=[A,B,C,D,E]`
`Cpv1=[87940,241600,1718,165400,798.7];`
`Cpv2=[29103 10040 2527 9356 1154];`
`Cpv3=[29105 8615 1702 103 910];`
`Cpv4=[33363 26790 2611 8896 1169];`
`Cpv5=[29370 34540 1428 26400 588];`
	The arrays defined are used to condense the list of coefficients from DIPPR. As mentioned in the comment in the first line of this row, they are in order as [A,B,C,D,E]. In MATLAB, an array of values uses an index system. In MATLAB, this index system always starts at a value of 1. For example, if you were to type disp(Cpv[1]), the value MATLAB displays would be 87940.
	
	(If you are familiar with other coding languages this may seem odd because in most other coding languages, 0 is typically the first index, but not in MATLAB)

```
Cp2=1/1e3*(Cpv2(1)*(T-To)+Cpv2(2)*Cpv2(3)*(1/tanh(Cpv2(3)/T)-
1/tanh(Cpv2(3)/To))-Cpv2(4)*Cpv2(5)*(tanh(Cpv2(5)/T)-tanh(Cpv2(5)/To)));

Cp3=1/1e3*(Cpv3(1)*(T-To)+Cpv3(2)*Cpv3(3)*(1/tanh(Cpv3(3)/T)-
1/tanh(Cpv3(3)/To))-Cpv3(4)*Cpv3(5)*(tanh(Cpv3(5)/T)-tanh(Cpv3(5)/To)));

Cp4=1/1e3*(Cpv4(1)*(T-To)+Cpv4(2)*Cpv4(3)*(1/tanh(Cpv4(3)/T)-
1/tanh(Cpv4(3)/To))-Cpv4(4)*Cpv4(5)*(tanh(Cpv4(5)/T)-tanh(Cpv4(5)/To)));

Cp5=1/1e3*(Cpv5(1)*(T-To)+Cpv5(2)*Cpv5(3)*(1/tanh(Cpv5(3)/T)-
1/tanh(Cpv5(3)/To))-Cpv5(4)*Cpv5(5)*(tanh(Cpv5(5)/T)-tanh(Cpv5(5)/To)));
```

The lines above are the integrated DIPPR Cp equations. As shown, each array value is called from its corresponding index value and array. Currently, there is no library that has the geometry function “coth”, this is substituted by 1/tanh(x) which is equal to coth(x).

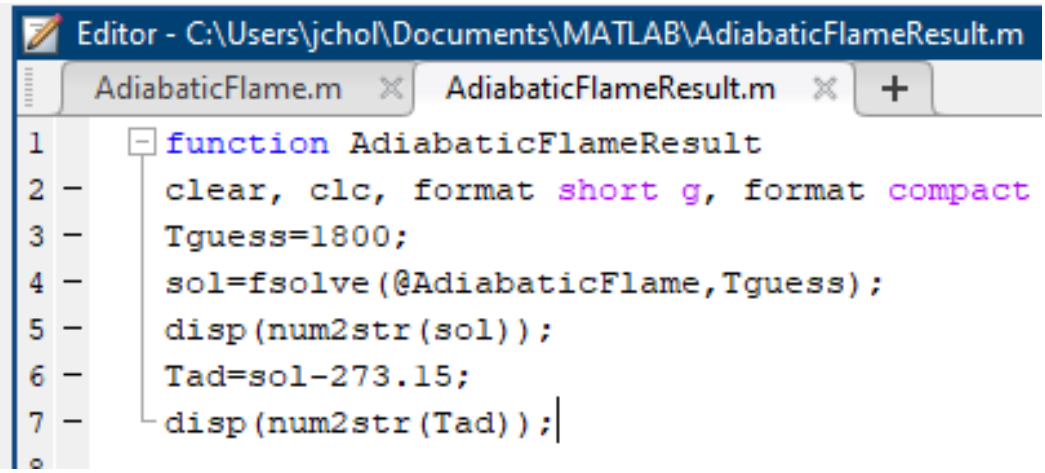
You may notice how the variable ‘T’ we are using here has not been defined yet. ‘T’ is the input parameter we talked about earlier. T is defined by the user when the function is called, unlike the other variables that will always be the value that they were assigned in the function

`H1Liq = H1formLq;`
`H2Vap = H2formVap + Cp2;`
`H3Vap = H3formVap + Cp3;`
`H4Vap = H4formVap + Cp4;`
`H5Vap = H5formVap + Cp5;`
	These are the total molar enthalpy values, calculated by adding the standard heats of formation/vaporization to the Cp.

`Hdotin = H1Liq * F1o;`
`Hdotout = H2Vap * F2 + H3Vap * F3 +`
`H5Vap * F5 + H4Vap * F4;`
	Inlet and outlet enthalpy flows, these were defined from the energy balance.

`fT = Hdotout - Hdotin;`	This is the energy balance, we know that for this problem that Inlet-Outlet=0 for steady state.
`end`	Tells MATLAB that all code above this and below the function declaration are a part of the function.

“AdiabaticFlameResult” Script



Shown above is the entire “AdiabaticFlameResult” script. Running this script will solve the problem using the equations and values that were defined in the previous script. Below is the explanation associated with this code. Keep in mind the name of this script and the other are both **case sensitive**. If they are misnamed, the functions that call on the “AdiabaticFlame” script will not be able to find the script because the names will be different.

`function AdiabaticFlameResult`	This initializes the function for this script. There are no input parameters associated with this function.
`clear, clc, format short g, format compact`	This line clears any previous memory the output had beforehand and formats it into a clear and concise output.
`Tguess=1800;`	This line is an initial guess for the temperature that is being solved for in the problem. Given that this is a combustion reaction, it is safe to assume the temperature is above the initial temperature of 298.15k
`sol=fsolve(@AdiabaticFlame,Tguess);`	This line calls on the function given in the “AdiabaticFlame” script using the Tguess shown above as the input parameter. This code will iterate until a solution is found, given it exists. An in depth explanation of the fsolve function can be found here: https://www.mathworks.com/help/optim/ug/fsolve.html
`disp(num2str(sol));`	Displays the solution to the problem in units of kelvin.
`Tad=sol-273.15;`	Calculates the solution in celsius
`disp(num2str(Tad));`	Displays the solution to the problem in celsius
`end`	Ends the function

Before running the code in the "AdiabaticFlameResult" script, make sure you save the code. After saving, press the run button found in the EDITOR tab to run the code for the solutions. Shown below is what the output should appear as after running.

```
Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the value of the function tolerance, and
the problem appears regular as measured by the gradient.

<stopping criteria details>
1996.5329
1723.3829
fx >>
```

Thus, the solution to the problem is 1996.5 K or 1723.4 degrees Celsius. You have now completed this MATLAB tutorial! If you have any comments or issues/problems, please don't hesitate to email chasen45@students.rowan.edu