<u>MATLAB Tutorial on ordinary differential equation solver (*Example 12-1*)</u>

Solve the following differential equation for co-current heat exchange case and plot *X*, *Xe*, *T*, *Ta*, and -*rA* down the length of the reactor (*Refer LEP 12-1, Elements of chemical reaction engineering, 5th edition*)

Differential equations

d(Ta)/d(V) = Ua*(T-Ta)/m/Cpc

d(X)/d(V) = -ra/Fa0

d(T)/d(V) = ((ra*dH)-Ua*(T-Ta))/Cpo/Fa0

Explicit equations

Cpc = 28
m = 500
Ua = 5000
Ca0 = 1.86
Fa0 = 14.67
dH = -34500
k = 31.1*exp((7906)*(T-360)/(T*360))
Kc = 3.03*exp((dH/8.314)*((T-333)/(T*333)))
Xe = Kc/(1+Kc)
ra = -k*Ca0*(1-(1+1/Kc)*X)
Cpo = 159

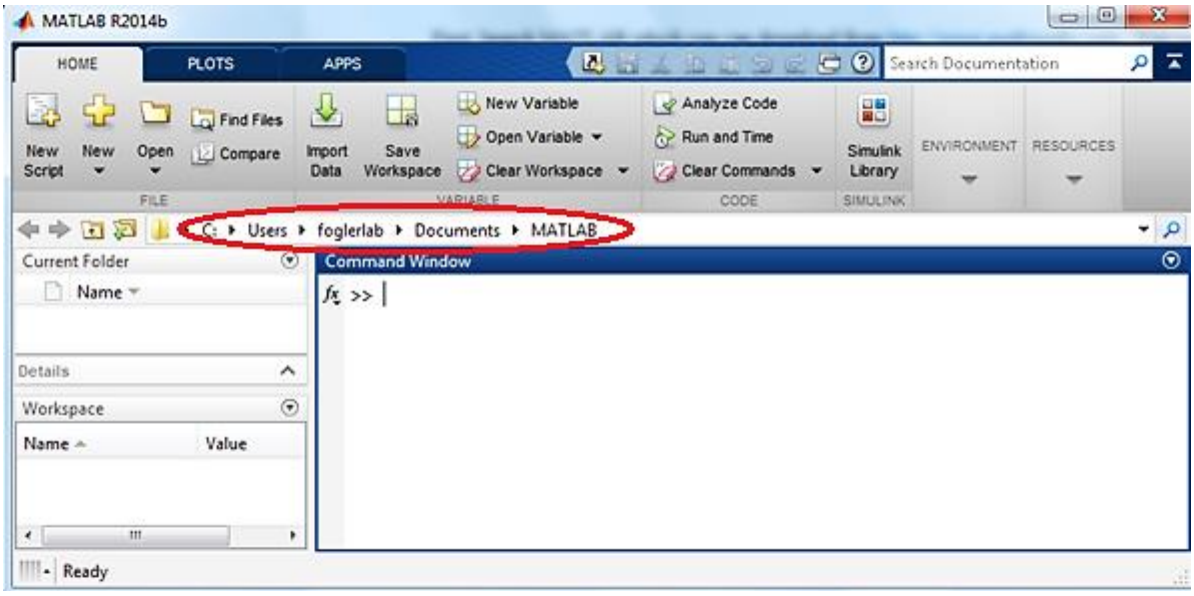
Initial and final values

Ta(0) =315
T(0)= 305
X(0)=0
V(0)=0
V(f)= 5

Repeat the similar exercise for other cases:

- Counter-current heat exchange: Plot *X*, *Xe*, *T*, *Ta*, and –*r*A down the length of the reactor
- Constant ambient temperature, *Ta*: Plot *X*, *Xe*, *T*, and –*r*A down the length of the reactor
- Adiabatic operation: Plot *X*, *Xe*, *T*, *Ta*, and –*r*A, down the length of the reactor

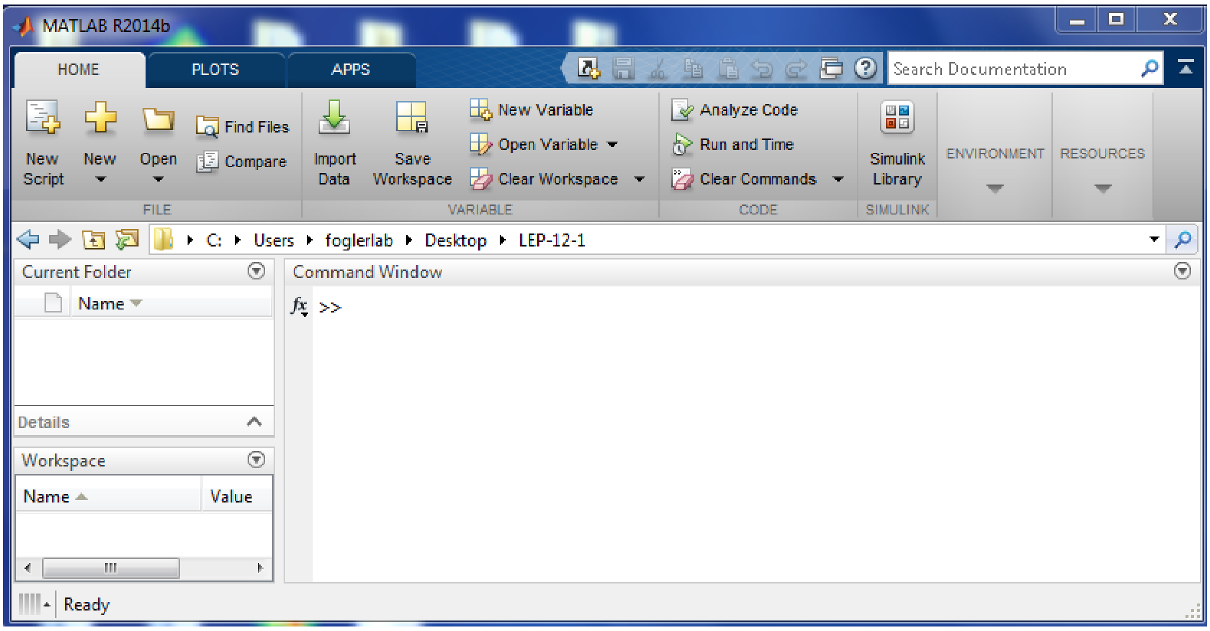
Step 1: First, launch MATLAB which you can access on CAEN computers (University of Michigan computer) or download from http://www.mathworks.com . You will see that multiple window opens which looks like this



The central window is called Command window. In the command window you can enter statements, run your files, generate output etc.

Step 2: Change the current folder location

The only folder which MATLAB can access is shown by red circle. In this case the folder name is "MATLAB". You must change the access location so that it refers to your particular folder which you are using for this project. Let's create a folder LEP-12-1 on desktop. Now change the MATLAB folder location to this location as shown below



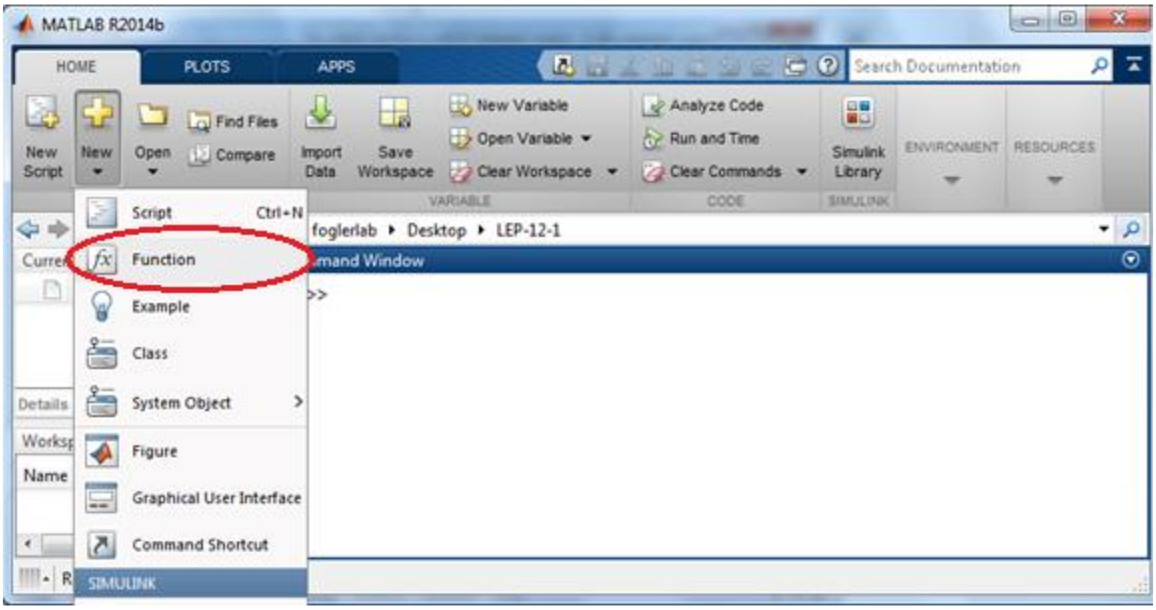
To solve ODE in MATLAB, you need to create two kind of program files:

1. **Script file** where you enter data such as integration span, initial guess, produce graphical outputs,etc
2. **Function file** where you enter all your explicit and differential equations

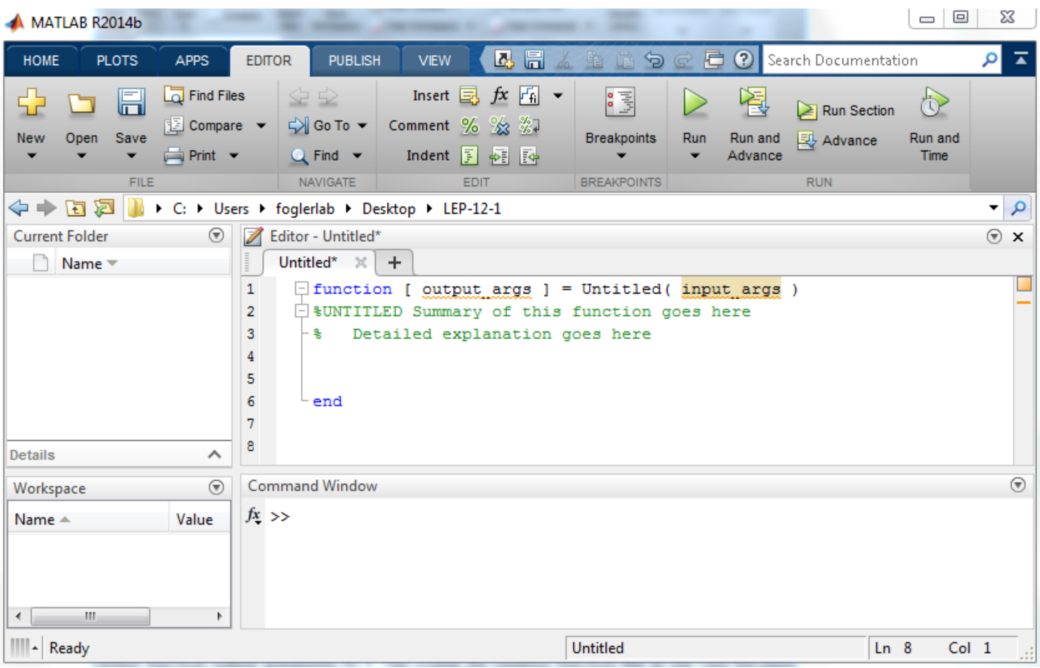
We will first create function file

<u>Creating function file</u>

Step 3: On the toolbar, Click on the New menu and select Function



You will see a new window opens that looks like this. MATLAB automatically creates syntax for writing function file. To use solver in MATLAB, you need to write codes in the space provided.

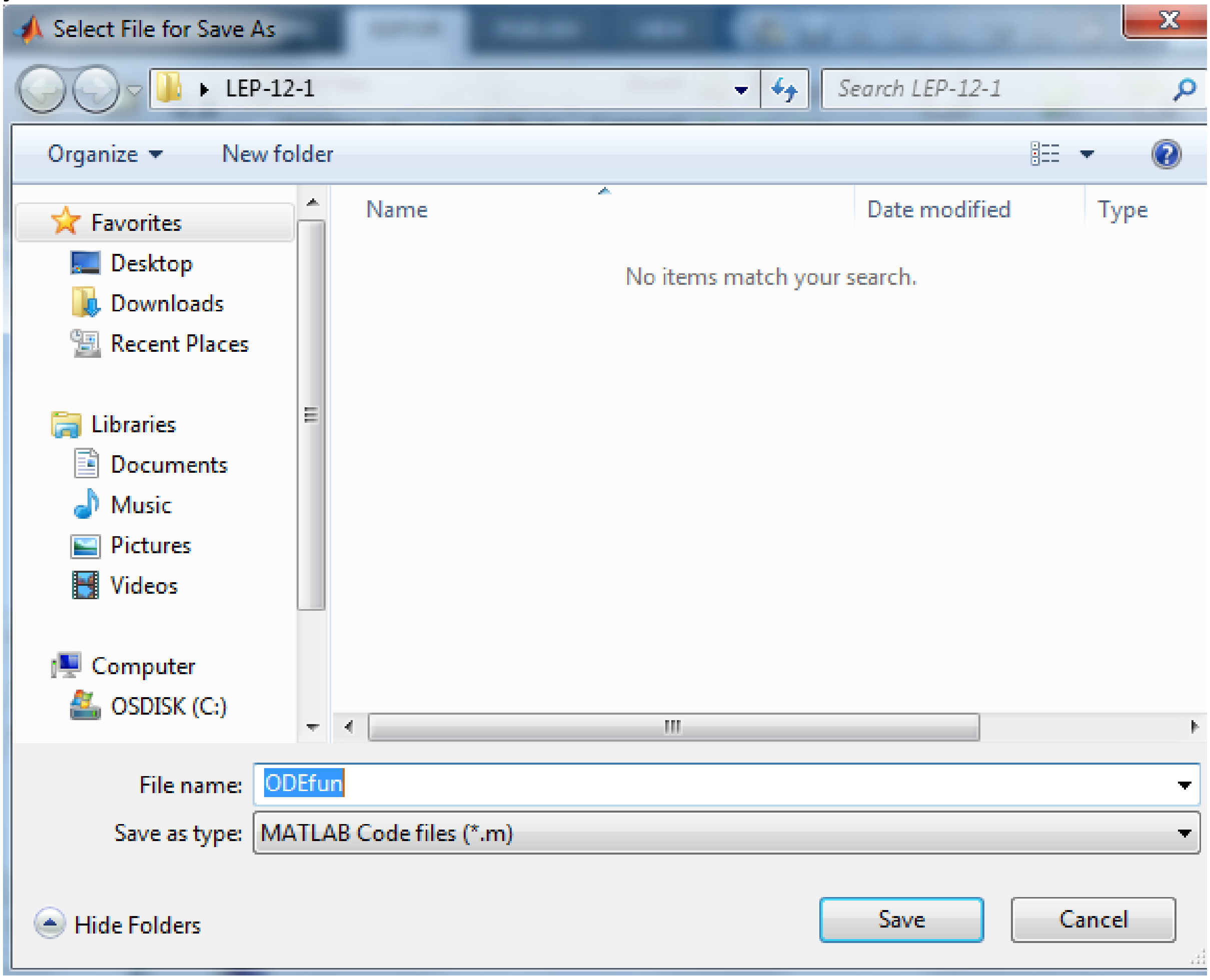


The first line of function starts with the keyword function followed by the output arguments. The right side contains function name (Untitled) and its input arguments. In this tutorial, we have chosen the function name as ODEfun which takes two input arguments i.e. V and Y. The first input argument “V” refers to integration span i.e. initial and final value of volume of reactor (in this case, Vinit=0, Vfinal=5).

Second input argument “Y” refers to initial values of the dependent variable i.e Ta, T, and X (in this case, Ta(0)=315, T(0)=305, and X(0)=0). The values of V and Y will be defined in the script file and then passed to the function file.

Step 4: SAVE your file.

Let’s name the function file as ODEfun. MATLAB file is saved with extension “.m”. In this case, your function file is saved with the name “ODEfun.m”



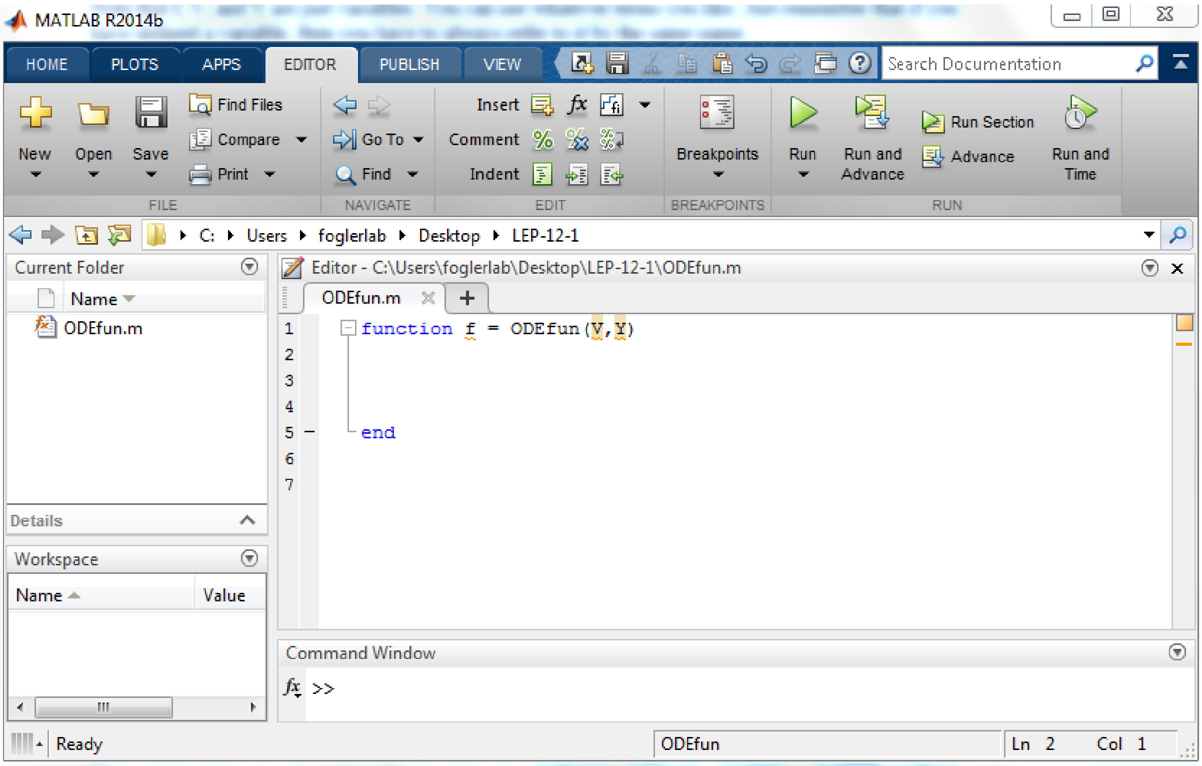
Step 5: Define function output arguments by f. The syntax for creating function file in our case becomes

Function f=ODEfun(V, Y)

Where V, Y are local to function.

Note that f, V, and Y are just variables. You can use whatever terms you like. Just remember that if you have defined a variable, then you have to always refer to it by the same name.

Now, edit the inbuilt format of function file for your case as shown below

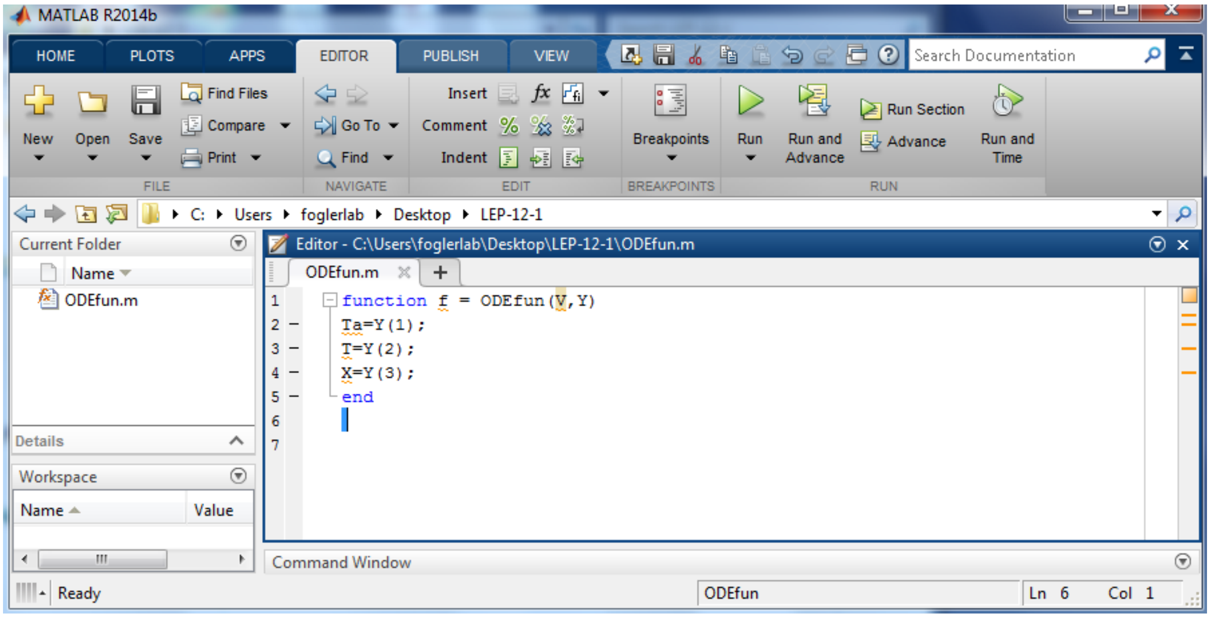


Step 6: As the differential equation contains 3 dependent variables (Ta, T, & X), so Y vector contains initial values of these 3 variables, where, Ta is the first element, T is the second element and X is the third element of Y vector

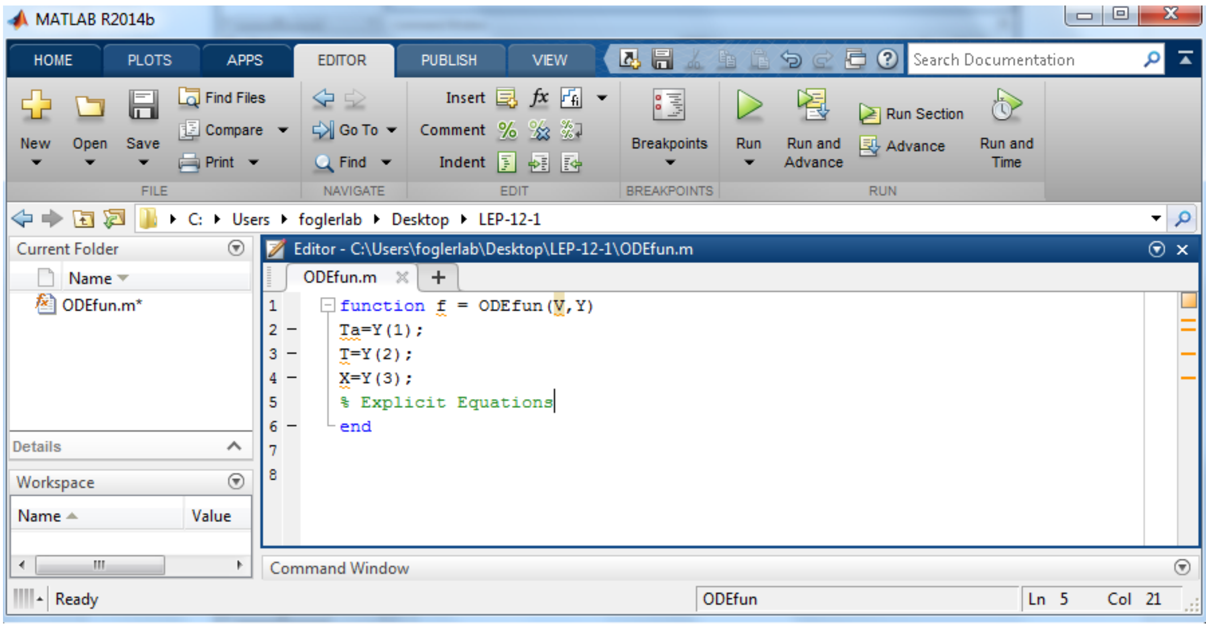
So,

Ta=Y(1), T= Y(2) and X=Y(3)

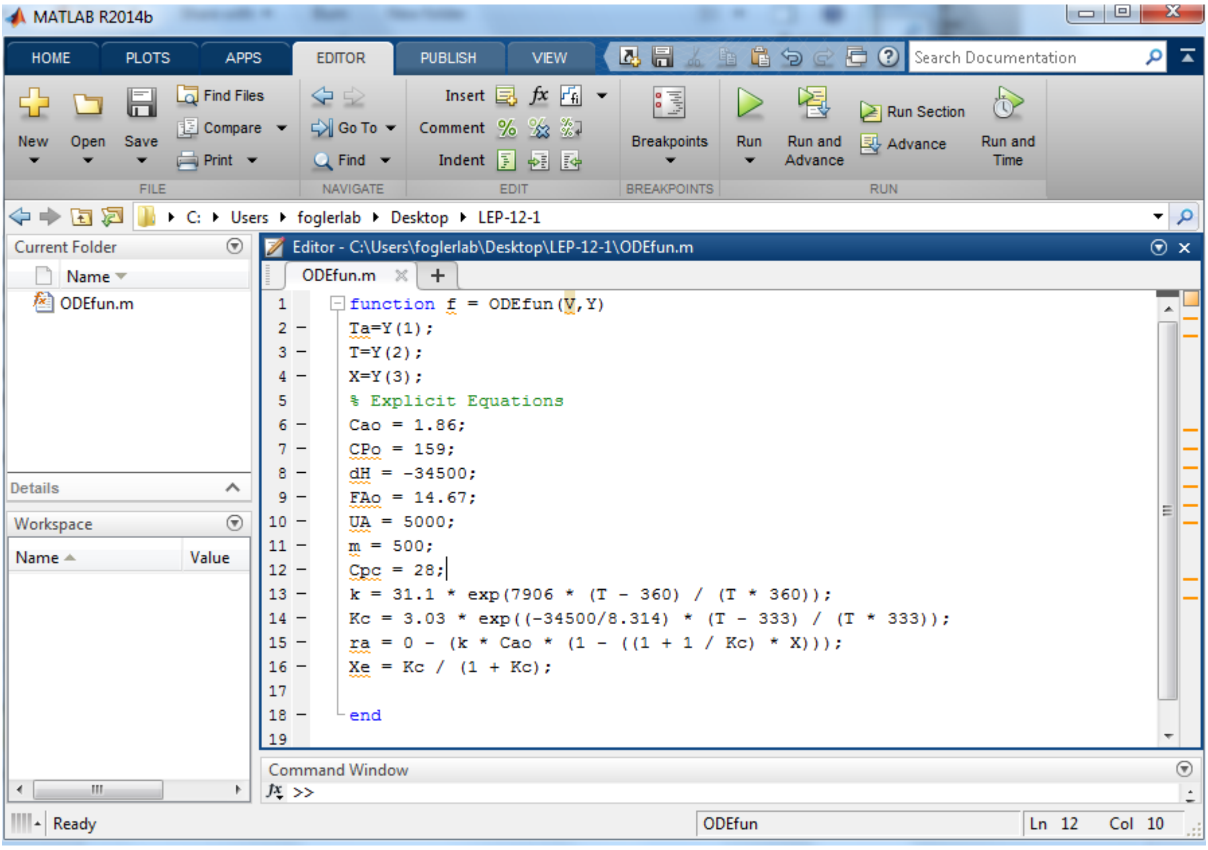
Assign the initial values of these variable as shown below. Put a semi-colon after each line to prevent the value from being displayed on the command window each time you run the file



Step 7: Before entering all the explicit equations, we will first write comments (which are not executed). To write comments, use percent symbol (%) followed by the comment. By default MATLAB uses green colour for writing comments. Let's put the comment "% Explicit equations"



Step 8: Now, enter all the explicit equations with semi colon at end



Step 9: Next, you need to enter your differential equations. For this example, you have three differential equation in Ta, T and X.

In MATLAB, LHS of differential equations cannot be entered in derivative form (dy/dx), so you need to define variable representing left side of differential equation

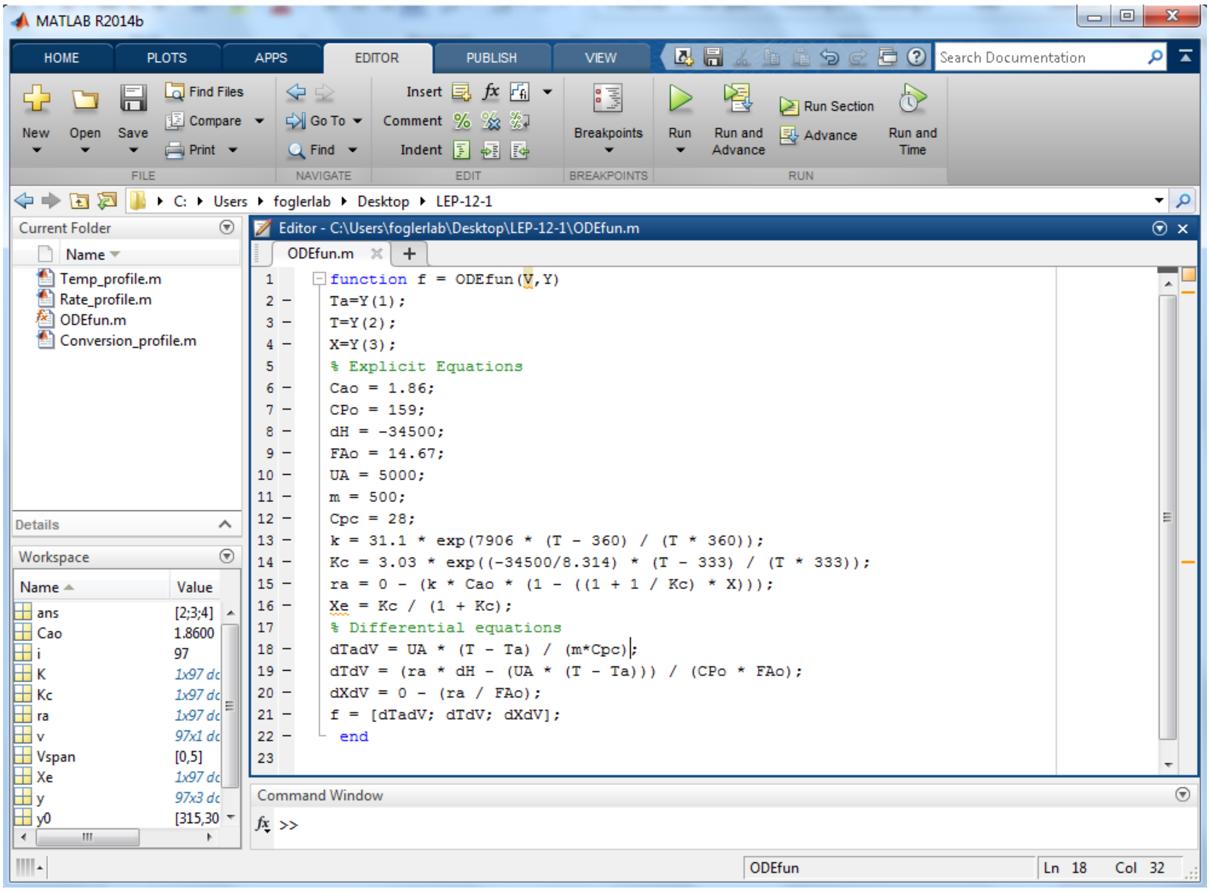
In this case we will use the following definition for differential equation

dTa/dV=dTadV,

dT/dV=dTdV, and

dX/dV=dXdV

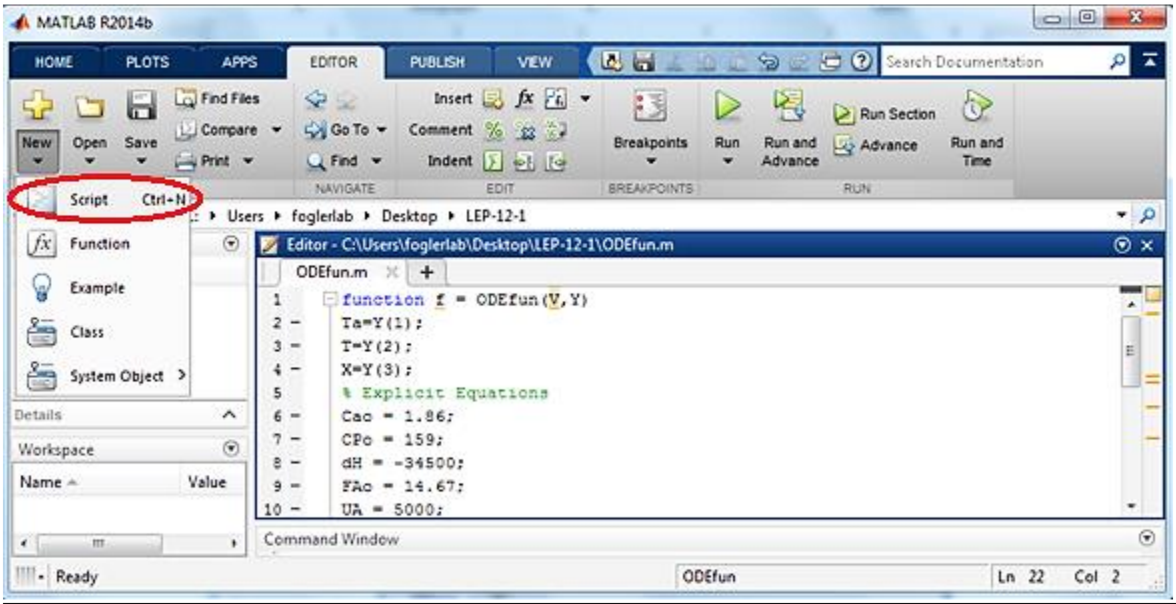
Enter the comment for differential equation and then enter your differential equations. After all the equations are entered, you need to define the output f. In the function file, f contains the differential equation. So, define f as shown below. ODEfun must return column vectors, so, you need to put semi-colon between differential equations to get column vector for different dependent variable.



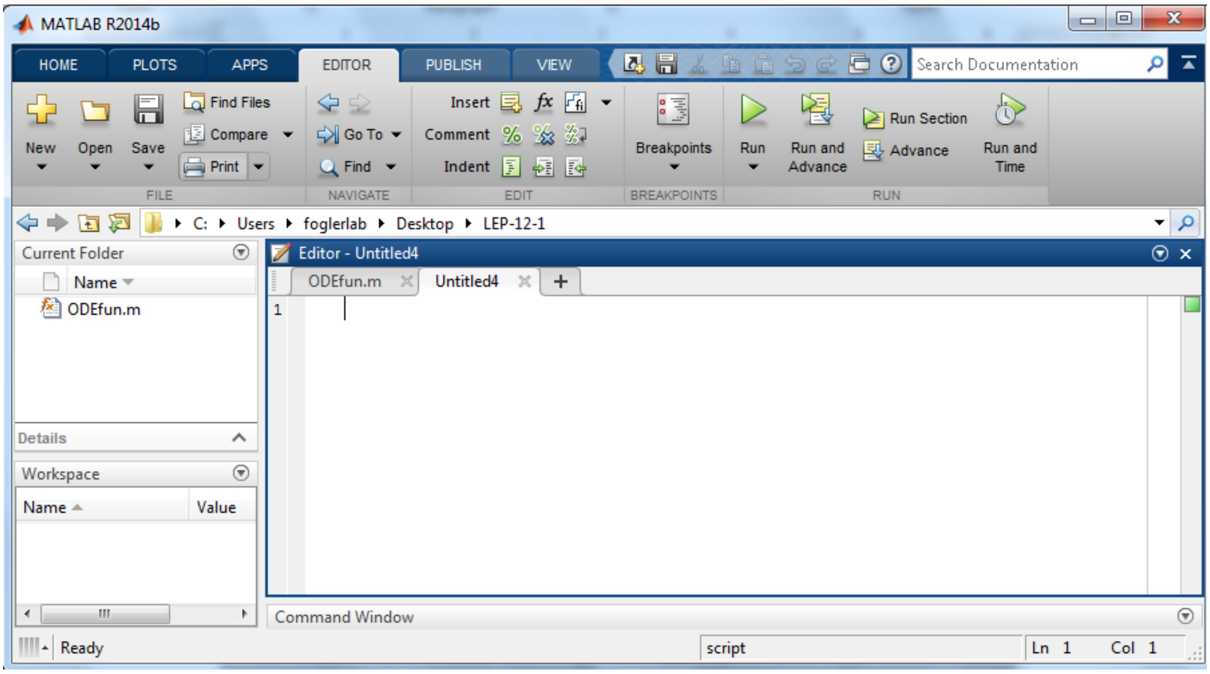
The function file returns the value in the form [v y] where, v is a column vector $\begin{bmatrix} V1 \\ V2 \\ Vn \end{bmatrix}$ of independent variable (i.e. volume for this case) and y is a matrix $\begin{bmatrix} Ta1 & T1 & X1 \\ Ta2 & T2 & X2 \\ Tan & T2n & Xn \end{bmatrix}$ of dependent variable (i.e. Ta, T, & X for this case). Note that no of rows are same in vector v and matrix y. The return value of the function will be used in the script file which would be discussed in next section

Creating a script file

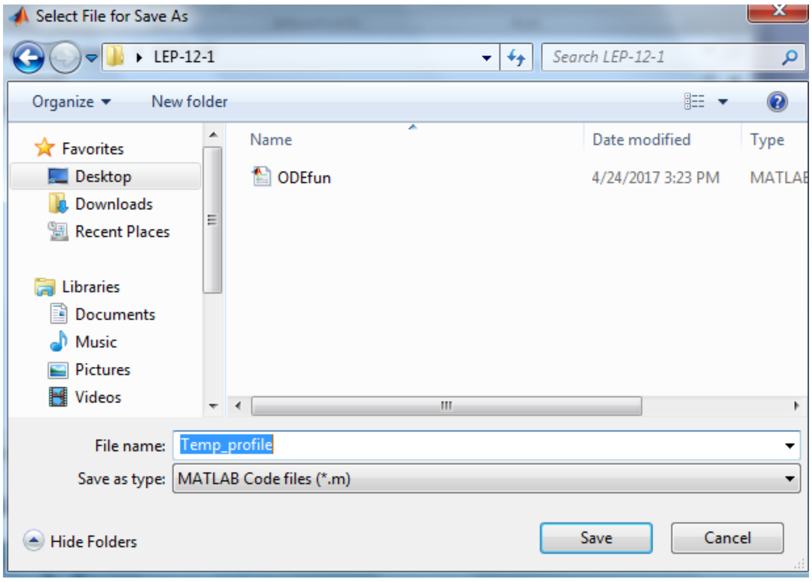
Step 10: Go back to New menu and select Script



A black window will appear like this



Step 11: First we will create the codes for Temperature profile. So save your script file with the name “Temp_profile”. You can also save it with other name as per your wish. We will save this file in the folder LEP-12-1 as shown



Step 12: In the blank space, Enter clc in the first line. It will clear all the input and output from the Command Window display, giving you a “clean screen”

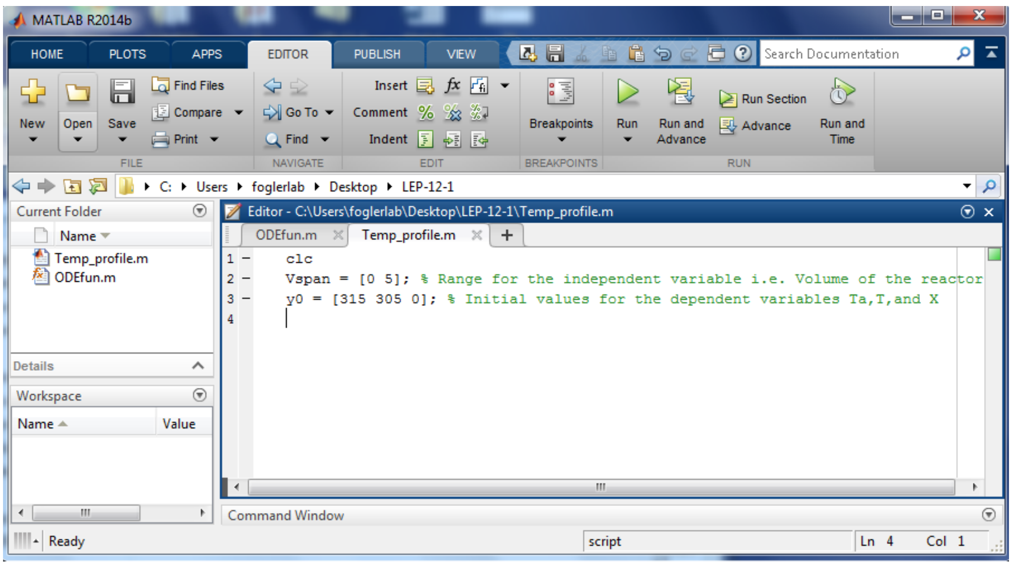
Next you need to enter the integration time span. In this case we want to integrate the volume of reactor from V=0 to V=5. Let’s define the integration time span variable as Vspan. To enter this in a row vector format, type “Vspan = [0 5]” with space between 0 & 5 else enter Vspan= [0 ; 5] to create a column vector. You can either create row or column vector, the output will remain same for this case. We will create a row vector.

Next you need to enter the initial values of the dependent variable, Ta, T, X i.e. Ta (0) =315, T (0) =305, and X (0) =0

Enter the initial value of the dependent variable in the vector form

y0= [315 305 0]

Again putting semi-colon at the end of each statement prevents the value from being displayed on the command window. We will also put comment against each line as shown below



Step 13: Next, you need to choose your ODE solver. There are different kind of solver available in MATLAB which you can use as per your problem requirement. The following is the list of all the solver with details:

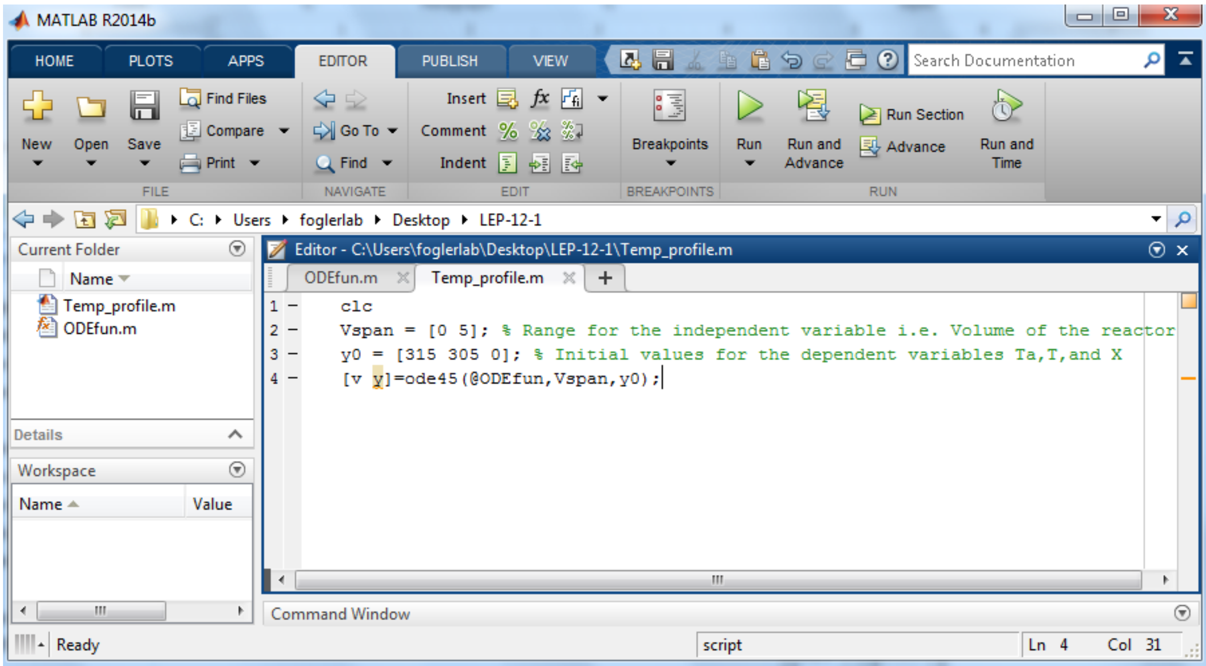
Solver	**Problem Type**	**Order of Accuracy**	**Method**	**When to Use**
ode45	Nonstiff	Medium	Explicit Runge-Kutta	Most of the time. This should be the first solver you try.
ode23	Nonstiff	Low	Explicit Runge-Kutta ,pair of Bogacki and Shampine	For problems with crude error tolerances or for solving moderately stiff problems.
ode113	Nonstiff	Low to high	Adams-Bashforth-Moulton PECE	For problems with stringent error tolerances or for solving computationally intensive problems
ode15s	Stiff	Low to medium	Numerical differentiation formulas (NDFs)	If ode45 is slow because the problem is stiff.
ode23s	Stiff	Low	Modified Rosenbrock	If using crude error tolerances to solve stiff systems and the mass matrix is constant.
ode23t	Moderately Stiff	Low	Trapezoidal rule using a "free" interpolant.	For moderately stiff problems if you need a solution without numerical damping
ode23tb	Stiff	Low	Implementation of TR-BDF2, an implicit Runge-Kutta formula with a first stage that is a trapezoidal rule step and a second stage that is a backward differentiation formula of order two	If using crude error tolerances to solve stiff systems.

The first choice for solving differential equation should be Ode45 as it performs well with most ODE problems. Hence, we will use ode45 solver. To use ODE solver, MATLAB uses following **Syntax**

[v y] = *solver* (@ODEfun, Vspan, y0)

Where ODEfun is the function file which you have created. The function file name must be same as that is invoked/called from the script file. Vspan is a vector specifying the interval of integration, and y0 is a vector of initial conditions

Step 14: Write down the solver equation in the required format as shown below. In the script file, we call/invoke function file and pass input arguments to function file. In this case input arguments are Vspan and y0.



When you run the script file, it will call function file and evaluate the differential equation for different values of independent variable and the output will be stored in [v y]. As described earlier, v is a column vector of volume and y is a column vector of [T, Ta, X]. We will not run the script file at this moment.

In this tutorial, you will learn to plot temperature profile, conversion profile and rate profile along the volume of the reactor.

a) Temperature profile

Step 15: The 'y' output of the function file contains value of Ta, T, and X where the value of Ta, T and X are in first, second column and third column respectively, so the values of these variables can be obtained as

Ta = y(:,1) ; T = y(:,2) ; X = y(:,3)   where, Ta, T and X are column vector

To plot Ta and T along the volume of the reactor, we will use MATLAB plot function.

The syntax for using plot function is

plot(X1,Y1,...,Xn,Yn)

Where X1, Y1 is the first set of data point. Similarly Xn, Yn is the nth set of data point. You can put multiple graph on the same plot by using comma between two data sets (X1, Y1) and (X2, Y2)

So to plot Ta vs v, the syntax is plot (v, y(: ,1))This will take the values of v (column vector) and 1^{st} column vector of y. To plot T vs v on the same graph, the syntax becomes

Plot (v, y(:1),v, y(:,2))

This will plot Ta and T on Y axis and v on the X axis. You may or may not put semi-colon at the end of Plot statement as this gives only graph and does not return any value on command window

We can also put legend, axis label, title, and range to your plot. The syntax are:

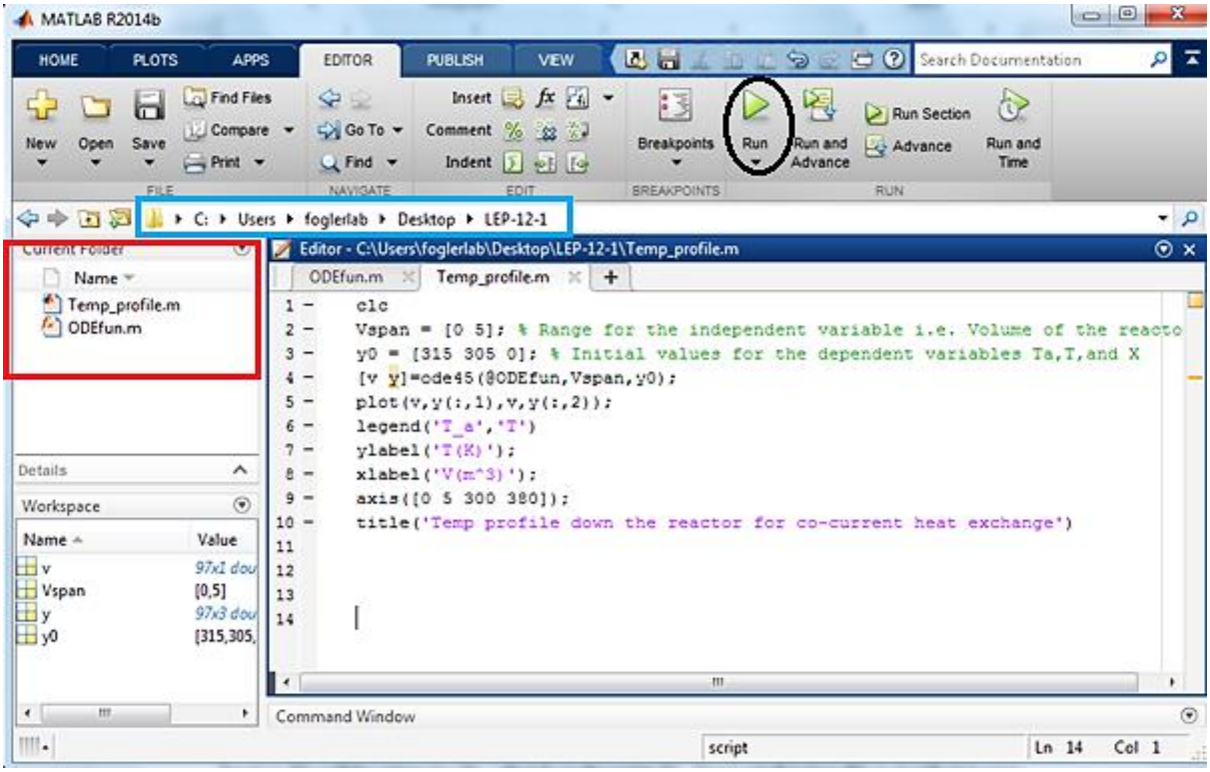
1) **For legend** : legend('comments', 'comments', 'comments')

   You can put your legend name under inverted comma. To put legend to different graphs, use comma between different legend names. By default, the order of the legend is same as the order of the graph defined in plot function

2) y axis label :ylabel('comments')
3) x axis label : xlabel('comments')
4) title: title('comments')
5) range: axis([a b c d]), where a, b refers to the range of X axis and c, d refers to the range of Y axis

The word in green can be replaced with the word you want to be displayed in your graph. In this case, there are two graphs: 1st is for Ta and 2nd for T. On X axis you want volume V (m3) and temperature T (K) on Y axis. The graph is made for co-current case, so accordingly define all the graphical features to your plot.

Enter the above codes as shown below

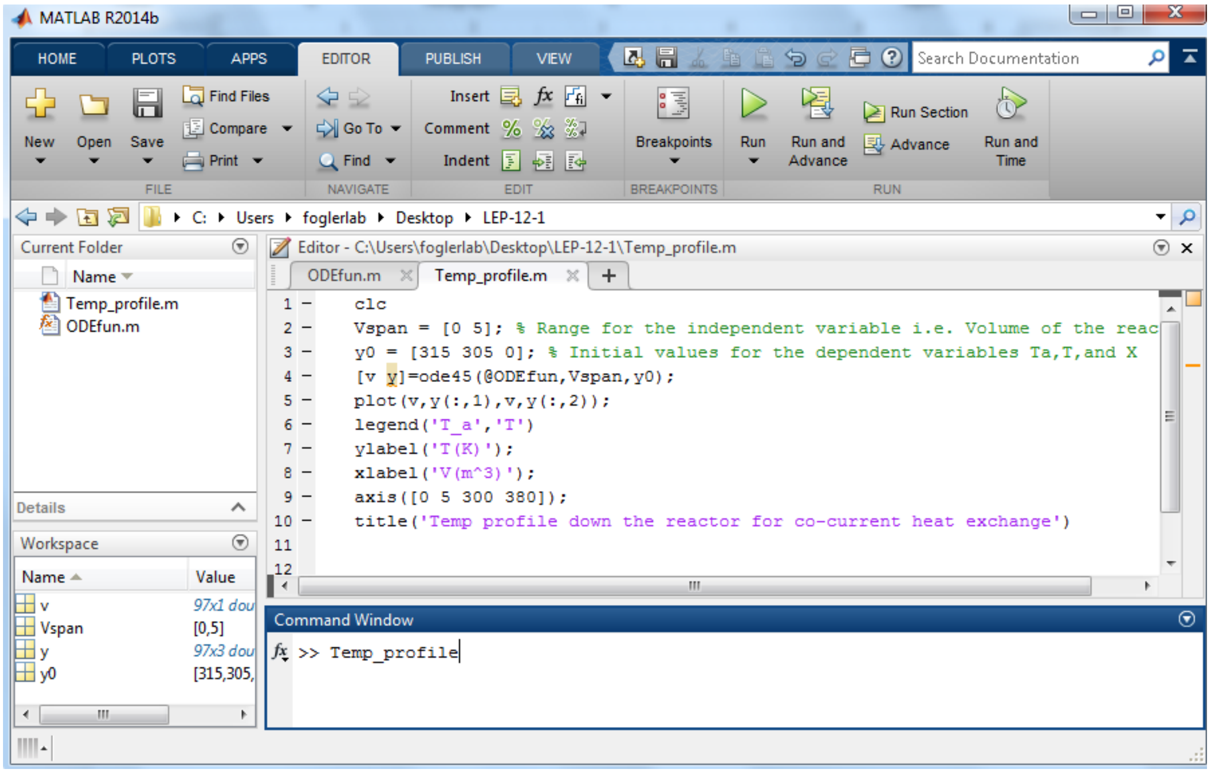


Now you have created both the script file and function file. You need to run only script file as the function file will be automatically called from the script file. Don't run function file as it will give error. This is because function file takes input arguments which are defined in the script file. Make sure that the function file is present in the same location as that of script file, else MATLAB won't be able to find the function file when you invoke the function file from script file.

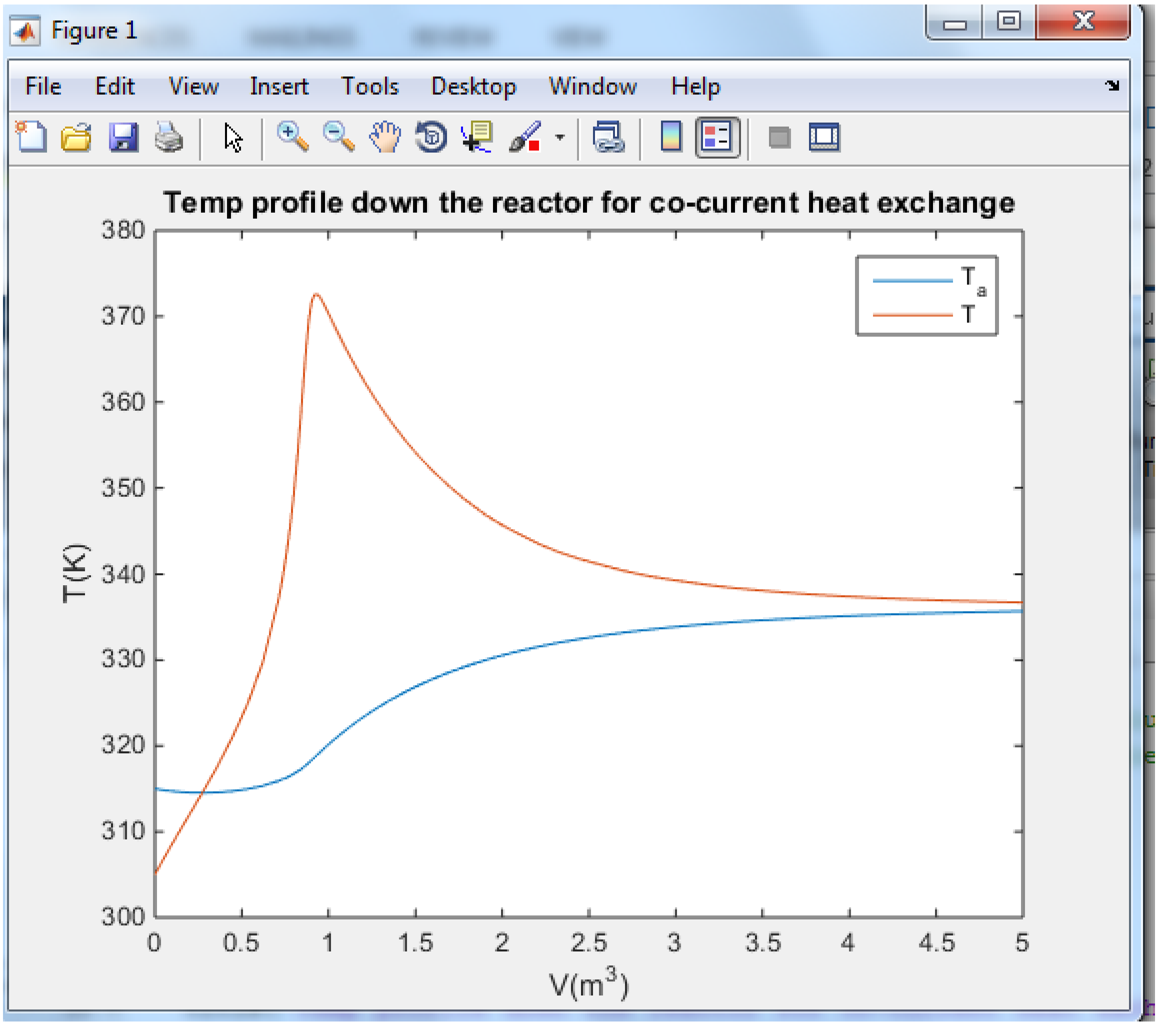
In this case, we have put both the files under the folder "LEP-12-1"

On the left side window, you will find that under current folder (shown by red rectangular box), both the files are present. If it is not so, then change the current folder location (shown by blue box) to the folder containing your files.

Step 16: In the Temp_profile file, press the run button  shown by black circle in the above screenshot. Alternatively, you can also run your script file by using the command window. In the command window type “Temp_profile and” press enter as shown below



You will see that following graph is generated which gives Ta and T profile down the length of the reactor.



You can also check that axis title, legend, chart title and axis range are as per defined by you in the code.

Next, you need to create conversion and rate profile along the length of the reactor

b) Conversion profile

Step 17: Create a new Script file and save it as "Conversion_profile" in the folder LEP-12-1

In this, you need to plot both actual conversion (X) and equilibrium conversion (Xe) along the length of the reactor.

Step 18: The function ODEfun gives conversion, X= y(:,3) as its output at different values of reactor volume. To get the value of Xe at different reactor volume you need to write few more codes.

The function file for this case will remain same as all the explicit equation and differential equations are going to be same. All you need to do is modify your script file to include the expression for Xe.

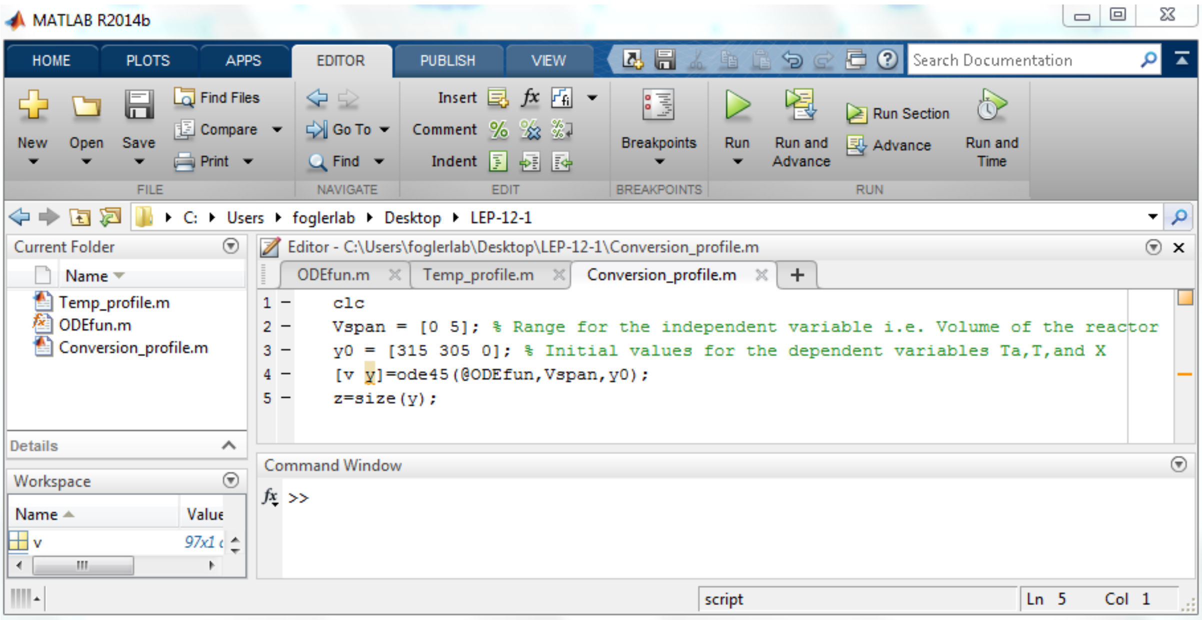
The function file will return the value of T at different values of v from which you can calculate value of Kc and Xe at different value of T. To get the total no of elements present in T vector, use MATLAB inbuilt "size" function which returns the size (no of rows and columns) of an array or matrix.

The values of temperature are stored in 1^{st} column of matrix y, so you need to find the size of y

The syntax for using size function is

Z=size (y); this will return the size of matrix y. Suppose size of y is n x m then Z= [n m]

Write the codes as shown below



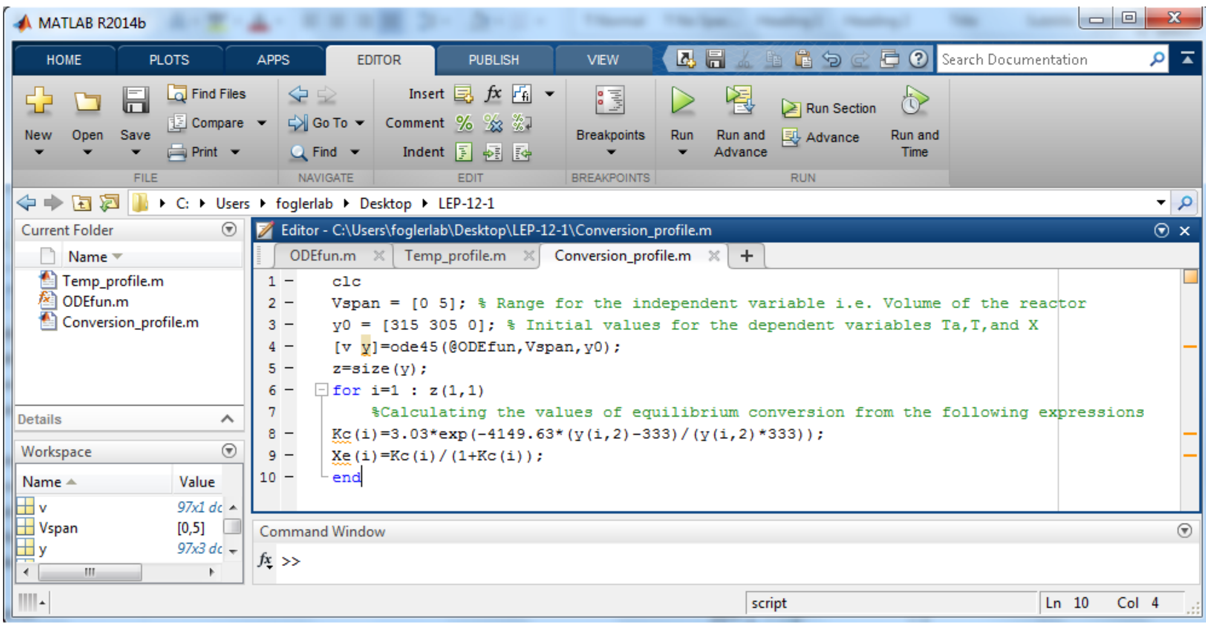
Step 19: We are only concerned with the row no of y i.e. z (1, 1)

Now Xe=Kc/(1+Kc)

And Kc = 3.03*exp((-34500/8.314)*((T-333)/(T*333)))

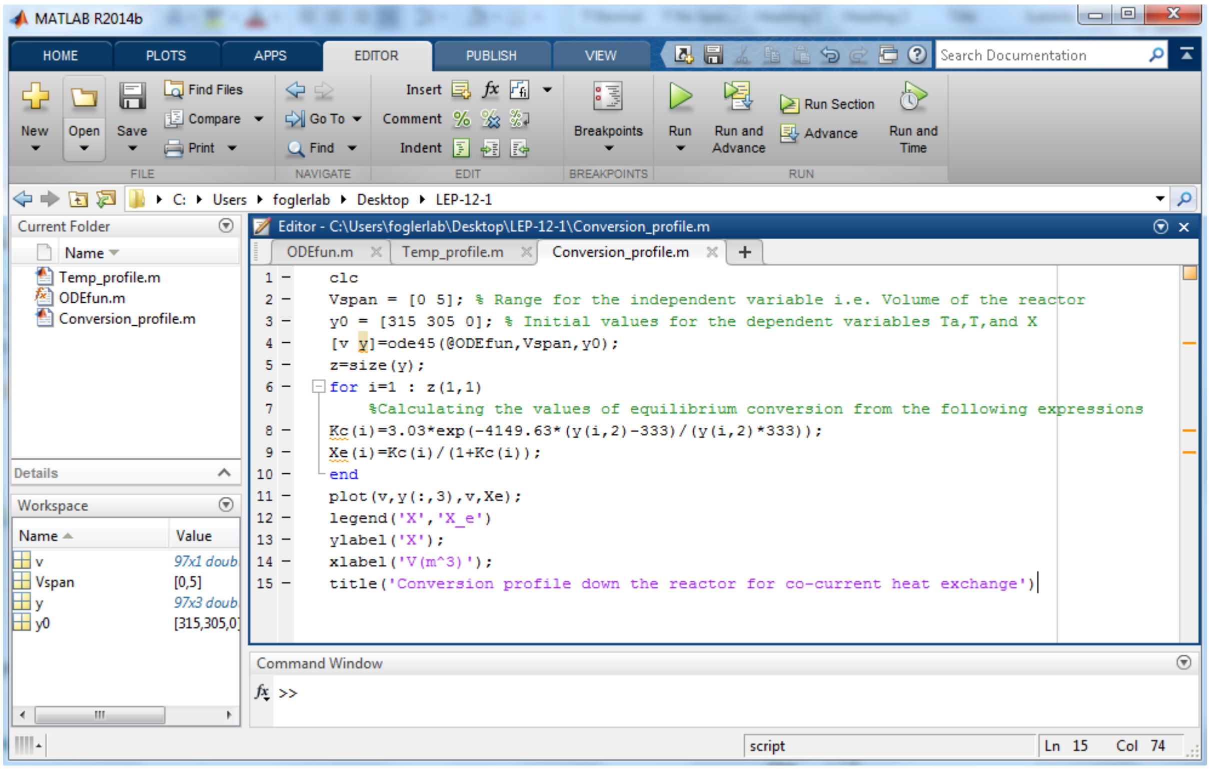
To evaluate the value of Xe at z (1,1) number of points, we need to create a "for" loop. We will first evaluate the value of Kc at different temperature and then calculate the value of Xe at different Kc. In the equation of Kc and Xe, T can be replaced by y(i, 2) as temp is the second dependent variable of y matrix

Now we will evaluate the value of Xe at i=1: z(1,1)

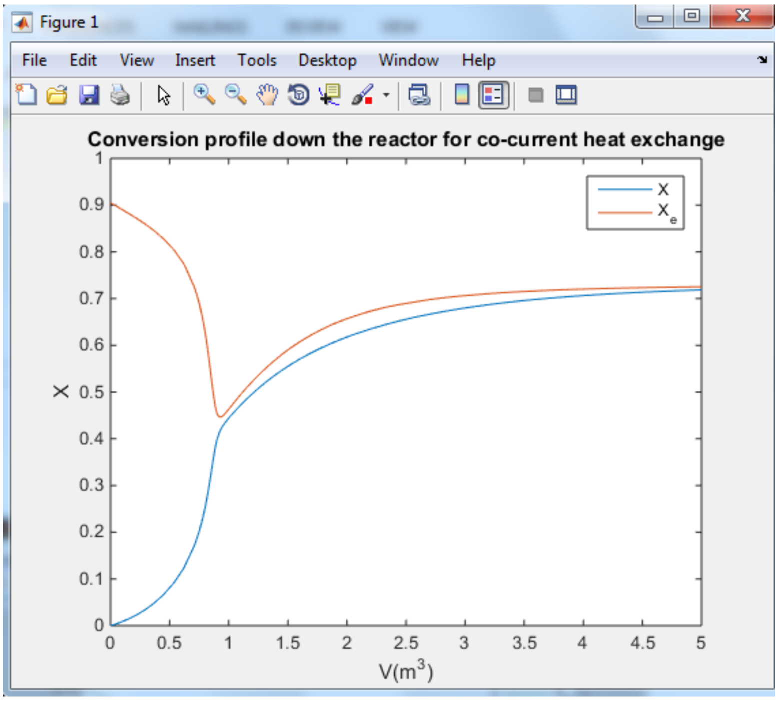


This will create a column vector of Xe with row no same as that of y matrix

Step 20: Next, you just need to add plot function as was done for case of temperature profile. The output y has X as the third element i.e. X=y(:,3). The vector Xe has been generated just now. So, write down plot function along with the graphical features. It is not necessary to specify axis range always as MATLAB can select the range automatically.



Step 21: Now run the conversion file either by clicking run button or typing "Conversion_profile" and pressing enter in command window. The following graph will be generated



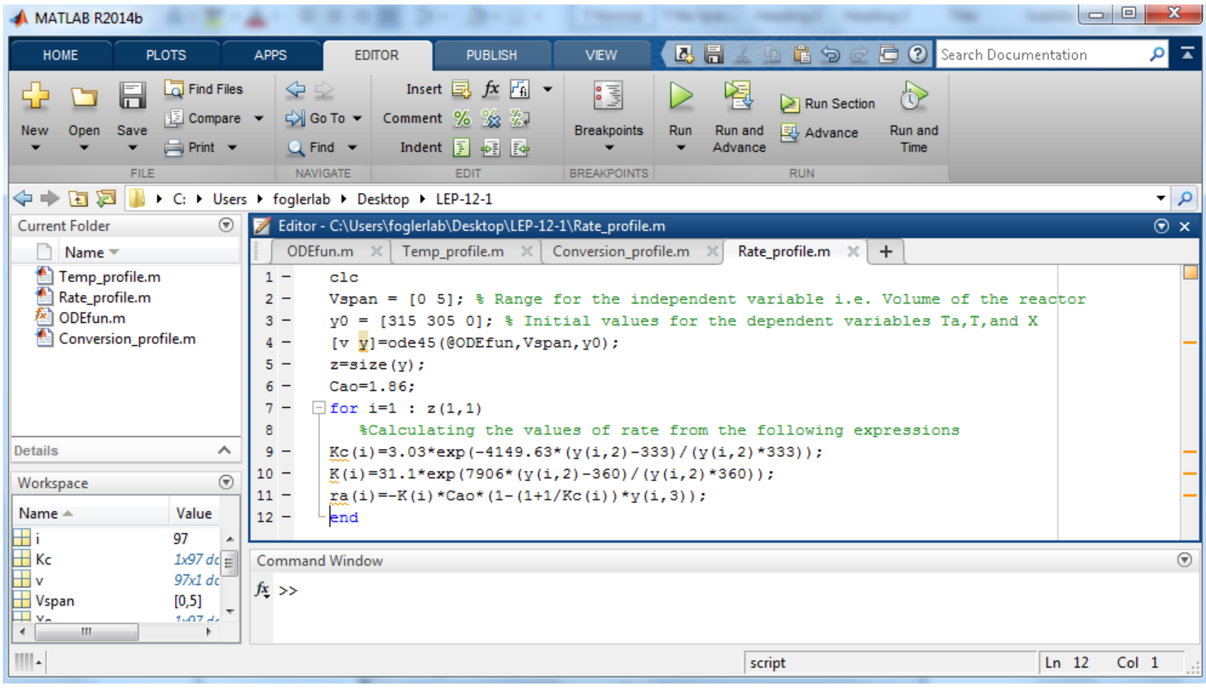
c) Rate profile

Step 22: Create a new Script file and save it as "Rate_profile" in the folder LEP-12-1

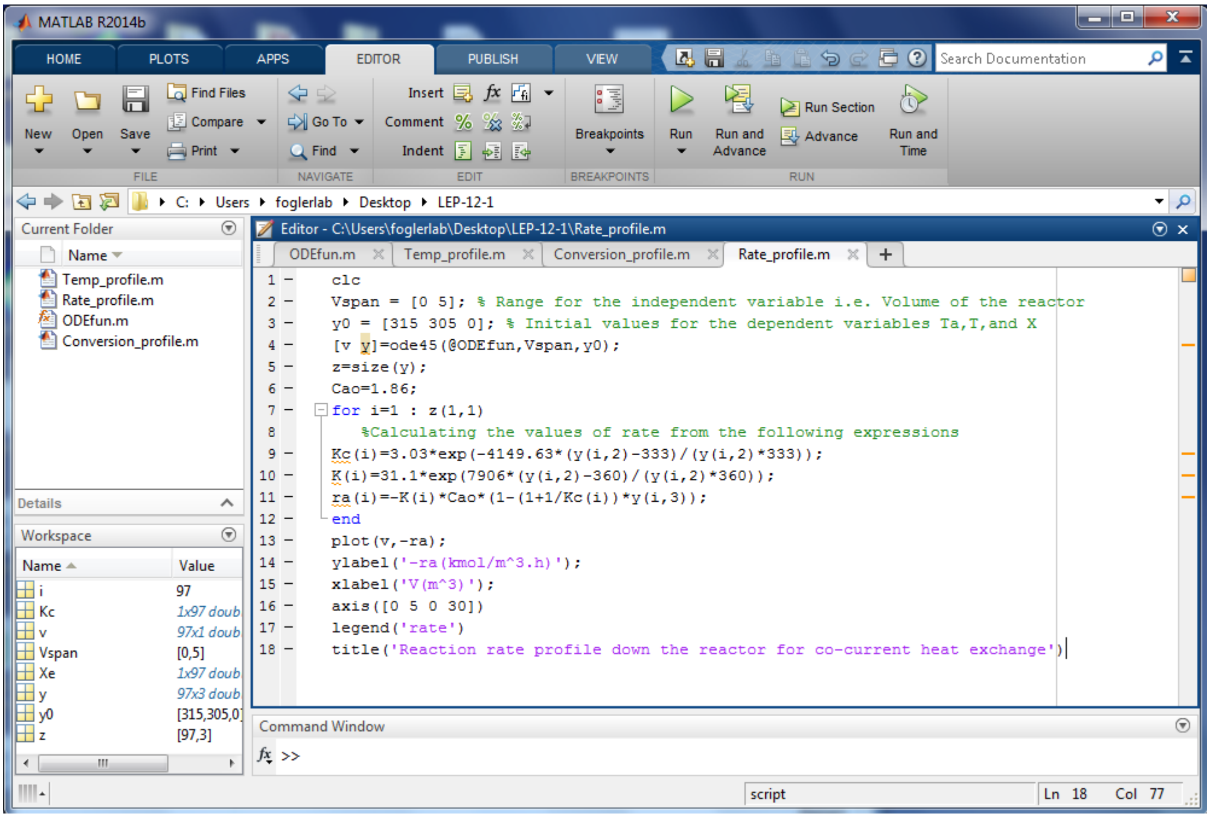
Step 23: For plotting rate profile, we need to determine the value of ra at different points. Again you need to edit only your script file. Create a "for" loop and replace T by y(i,2) and X by y(i,3) in the expression of k, Kc, and rate. So the equation for rate becomes:

k(i) = 31.1*exp((7906)*(y(i,2)-360)/(y(i,2)*360))
Kc(i) = 3.03*exp((-34500/8.314)*((y(i,2)-333)/(y(i,2)*333)))
ra(i) = -k(i)*Cao*(1-(1+1/Kc(i))*y(i,3))

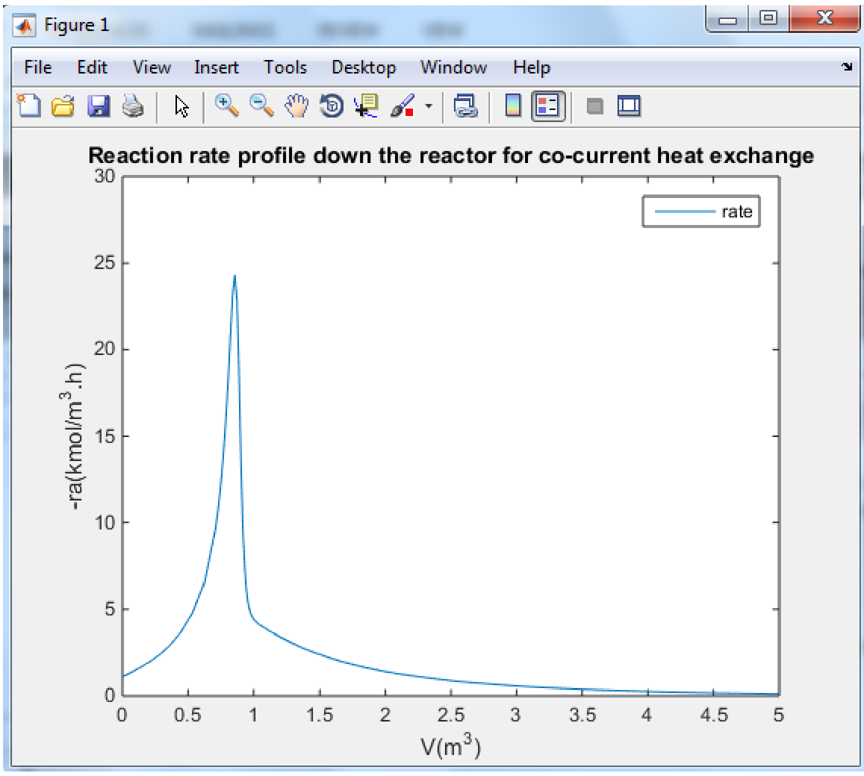
Insert the codes in the file as shown below. This will create a column vector of k, Kc and ra



Step 24: Next, add plot function, label your axis, define the range, legend and title on the graph. If the graph doesn't fit it the axis range you have provided, then go back to file and re-set the axis range. You need to plot rate function which is negative of ra (rate= - ra). So, in the plot function '–ra' is used to plot rate.



Step 25: Now run the file. You will get an output that looks like this

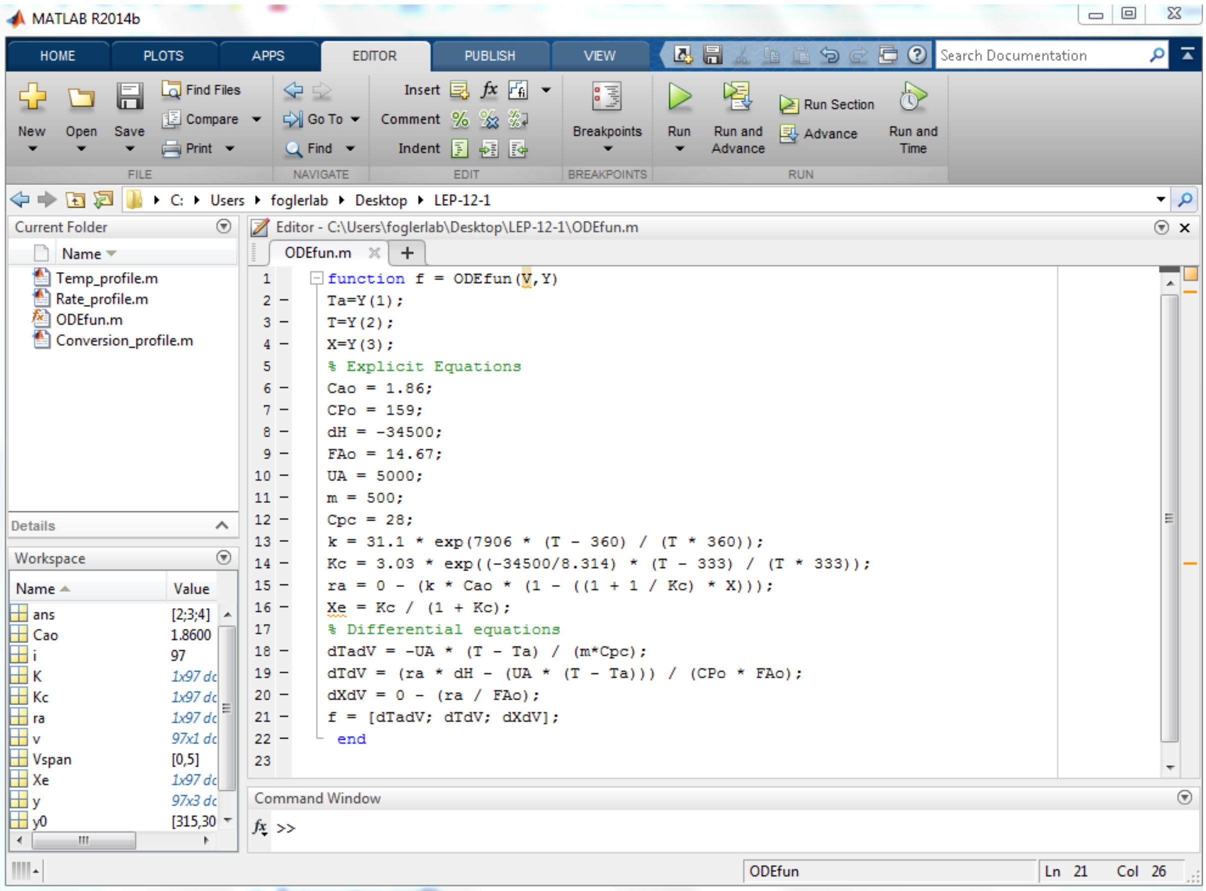


Counter- current heat exchange

Step 26: For counter-current flow, we only need to make two changes in the program. First, we modify the expression for Ta for counter-current flow. Multiply the right hand side of differential equation for Ta with -1. The following equation is obtained

$$d(Ta)/d(V) = -Ua*(T-Ta)/m/Cpc$$

To modify equation for Ta, open the function file ODEfun and put a minus sign on the right hand side of Ta equation as shown below:



Step 27: Save your file.

Step 28: Next, we guess *Ta* at *V* = 0 and see if it matches *Tao=315* at *V* = 5 m3. If it doesn't, we guess again. In this example, we will make first guess *Ta* (*V* = 0) = 330 K and see if *Ta* = *Ta*0 = 315 K at *V* = 5 m3.

Change the initial value of y0 in all the script file as described below

a) <u>Temperature profile</u>

To make the changes, open your script file "Temp_profile" and change the 1st guess value in y0 to 330 instead of 315 as shown below. The value of Ta at the end of reactor will be the last element of 1st column vector of y. So find the size of y and evaluate the value of y(n,1), where n is the last point

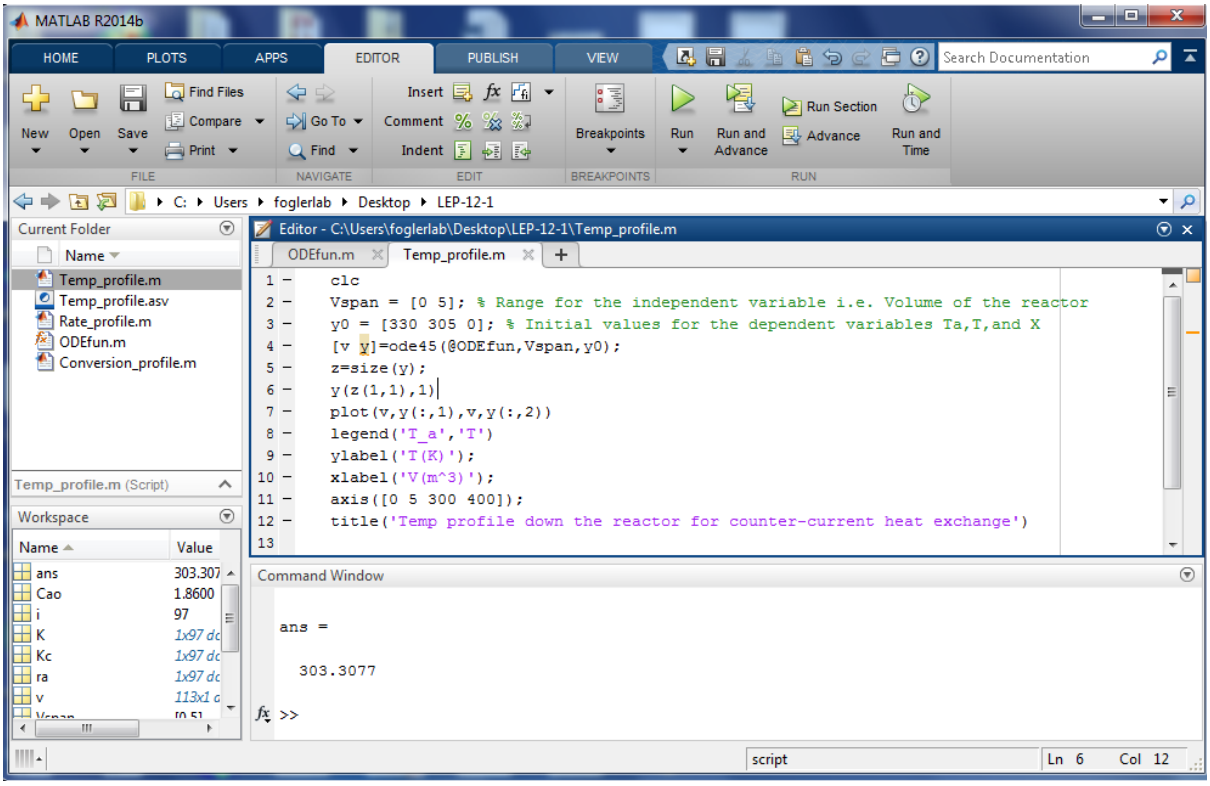
So,

z=size(y);

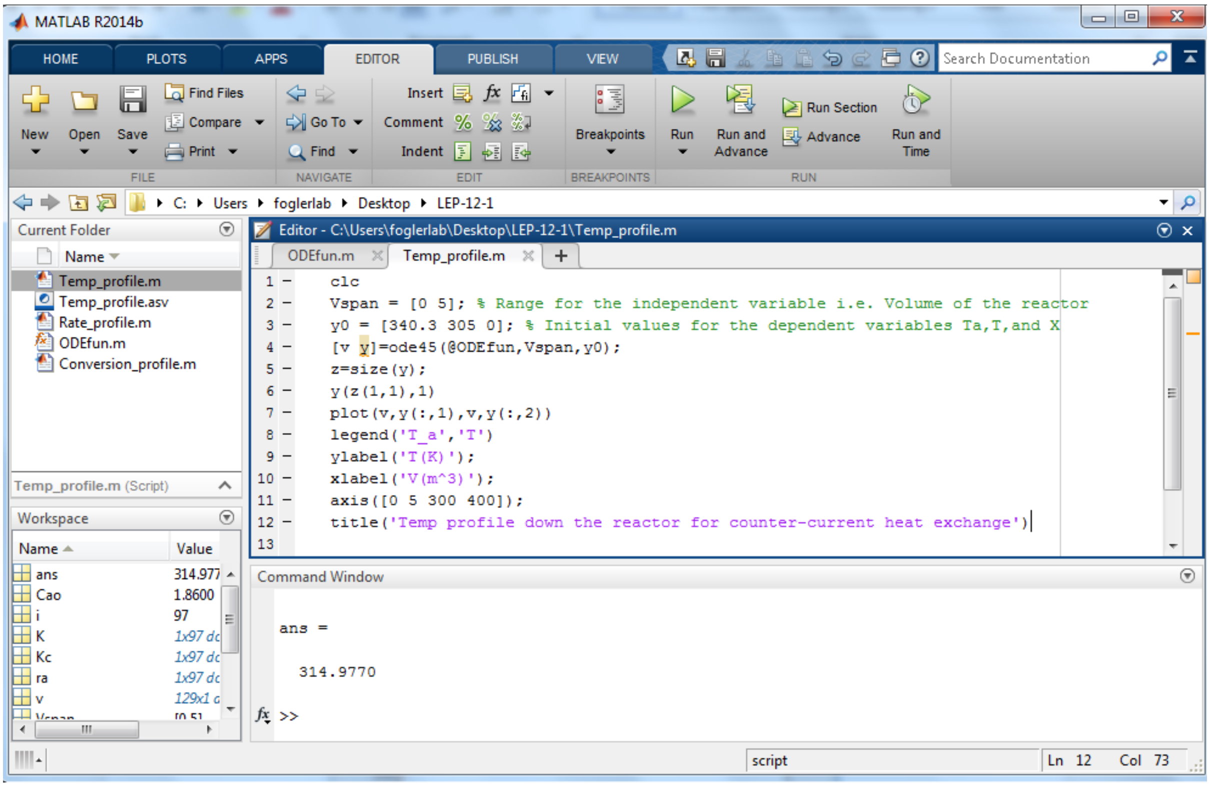
The value of Ta at the end of reactor will be given by
y(z(1,1),1), where z(1,1) gives the row number of last element

Don't put semi-colon at the end of above line as you want the value of Ta to be displayed on command window. Also change the title of the graph from co-current to counter-current.

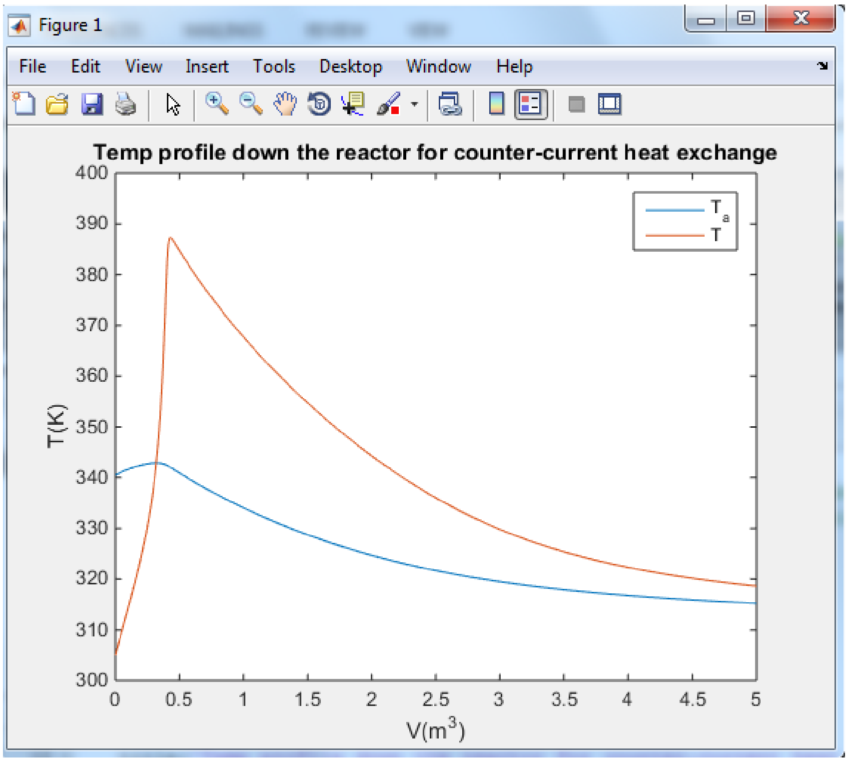
Step 29: Now save your file and run the program. In the command window, you can see that outlet temperature of Ta is 303.3 K but you want Ta=315 K



Step 30: Make another guess and check the Value of Ta. The final guess we obtain is Ta (V)=340.3 K where Ta (0)=315 K

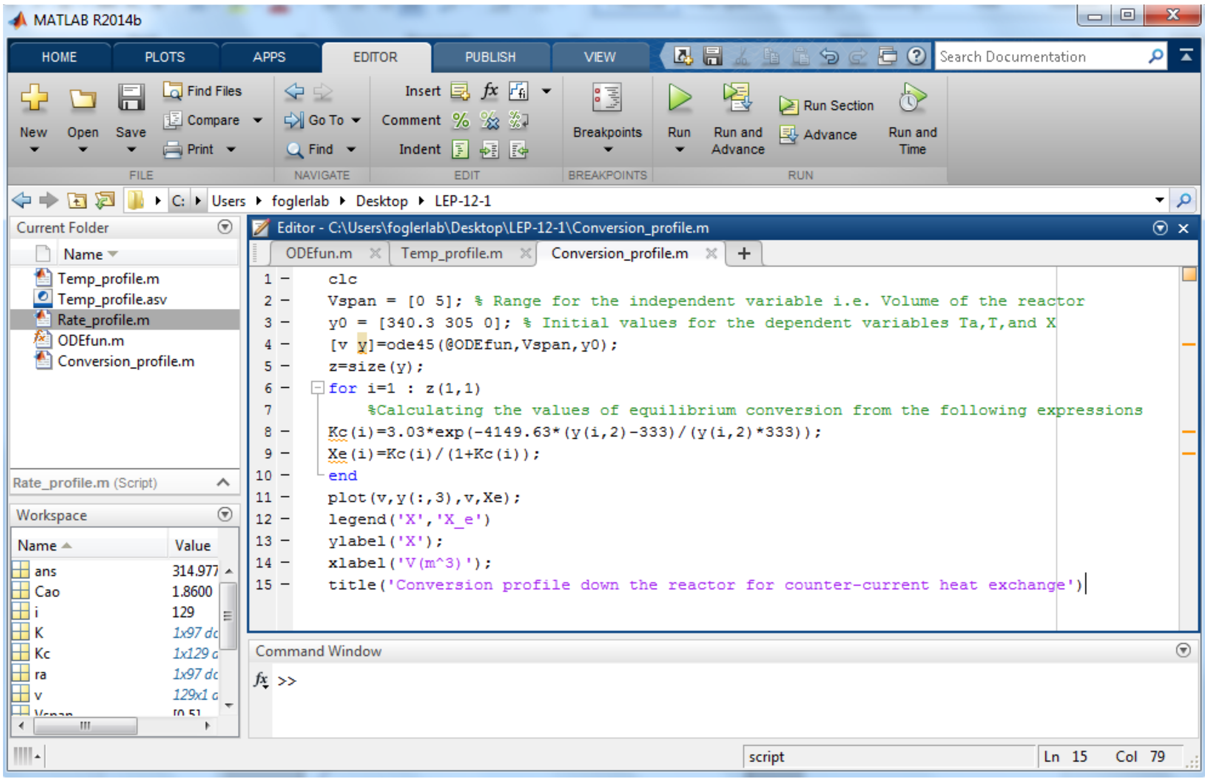


The following graph is obtained

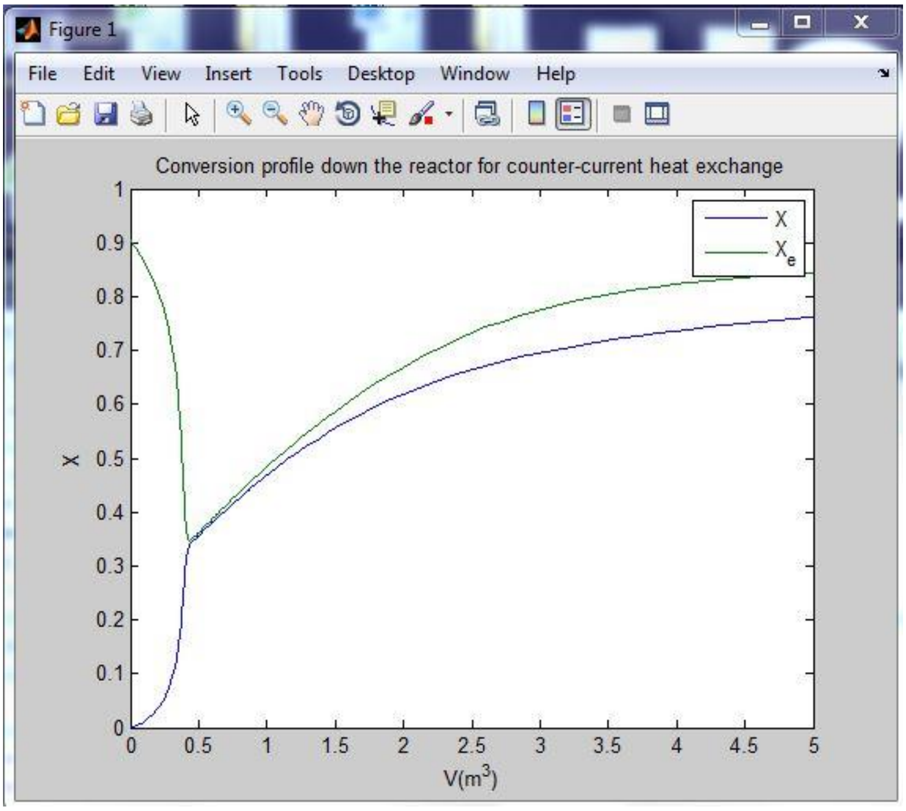


b) Conversion profile

Step 31: Change the initial value of Ta (i.e. Ta=340.3) and graph title in “Conversion_profile.m” file as shown

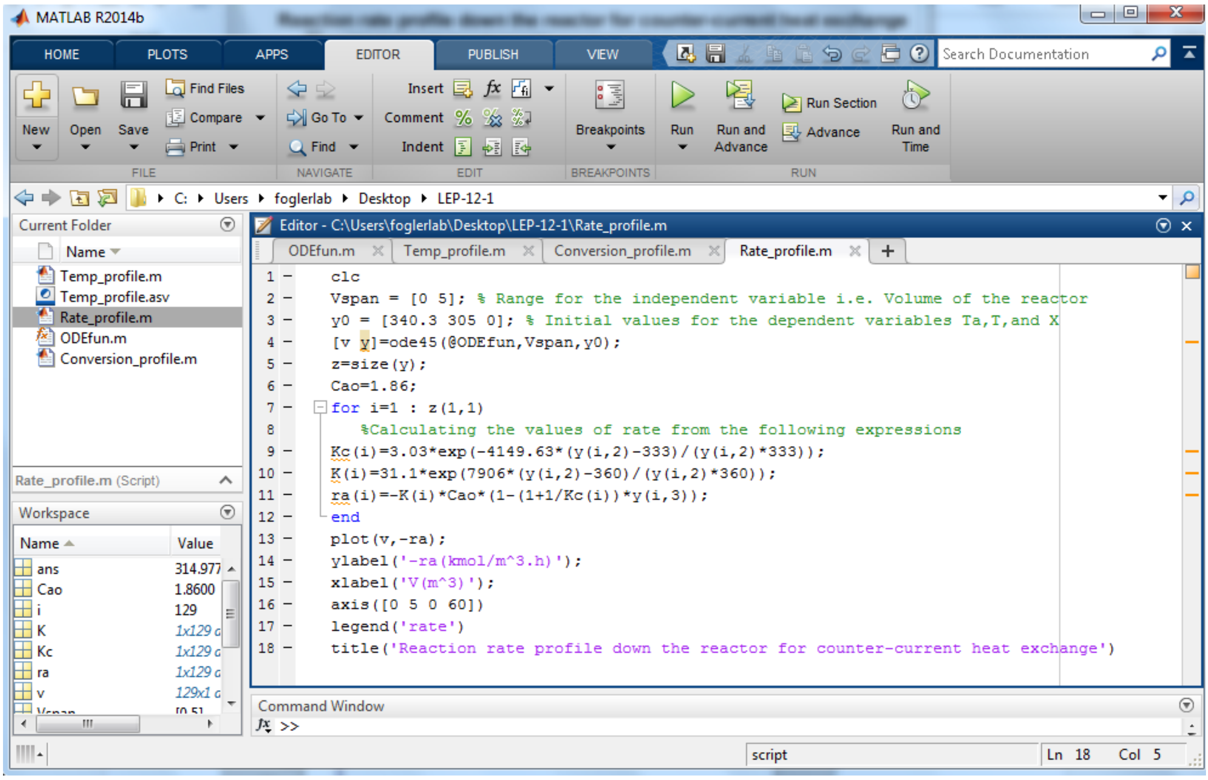


When you run the program, you will get an output that looks like this

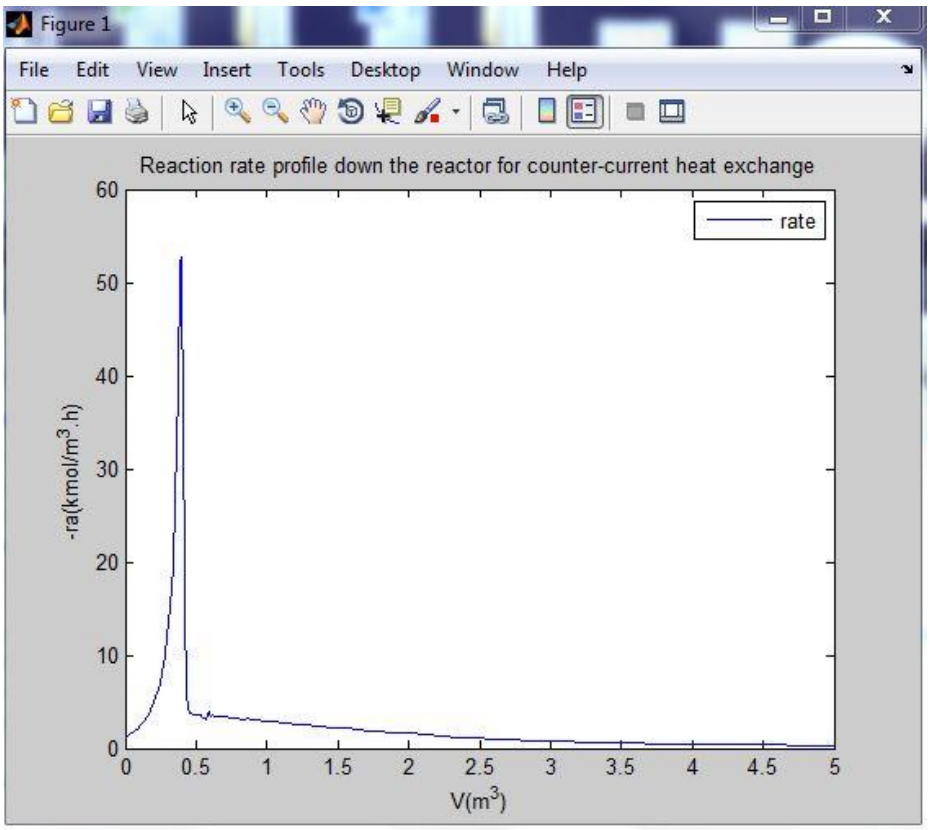


c) Rate profile

Step 32: Change the initial value of Ta and graph title in rate profile as shown



You will get an output like this

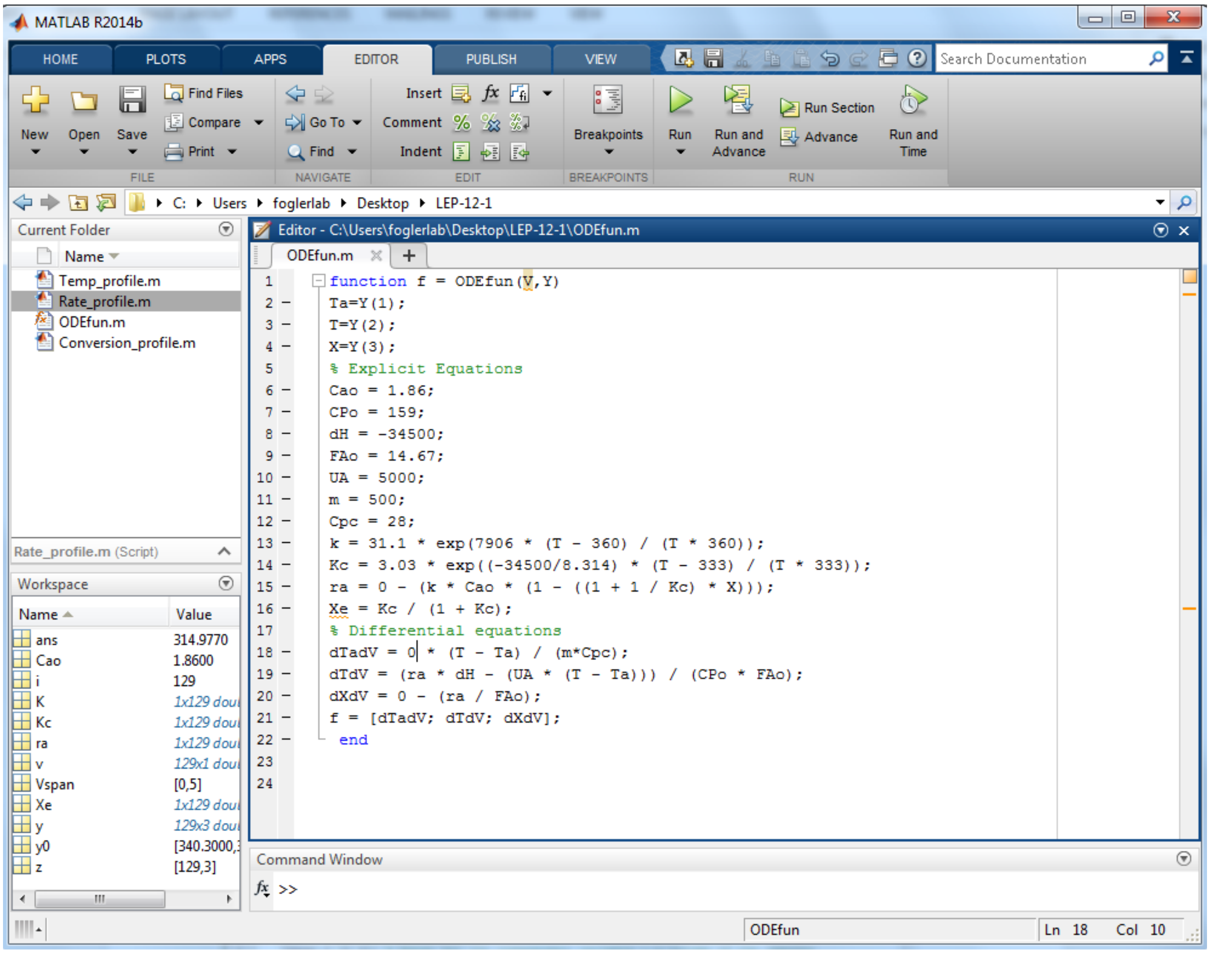


Constant Ta case

Step 33: For Constant Ta, we only need to make one change i.e. modify the expression for Ta. Multiply the right hand side of differential equation for Ta with 0. The following equation is obtained

d(Ta)/d(V) = 0 *Ua*(T-Ta)/m/Cpc

In the function file, modify the equation of Ta only. All other equations will remain as it is.

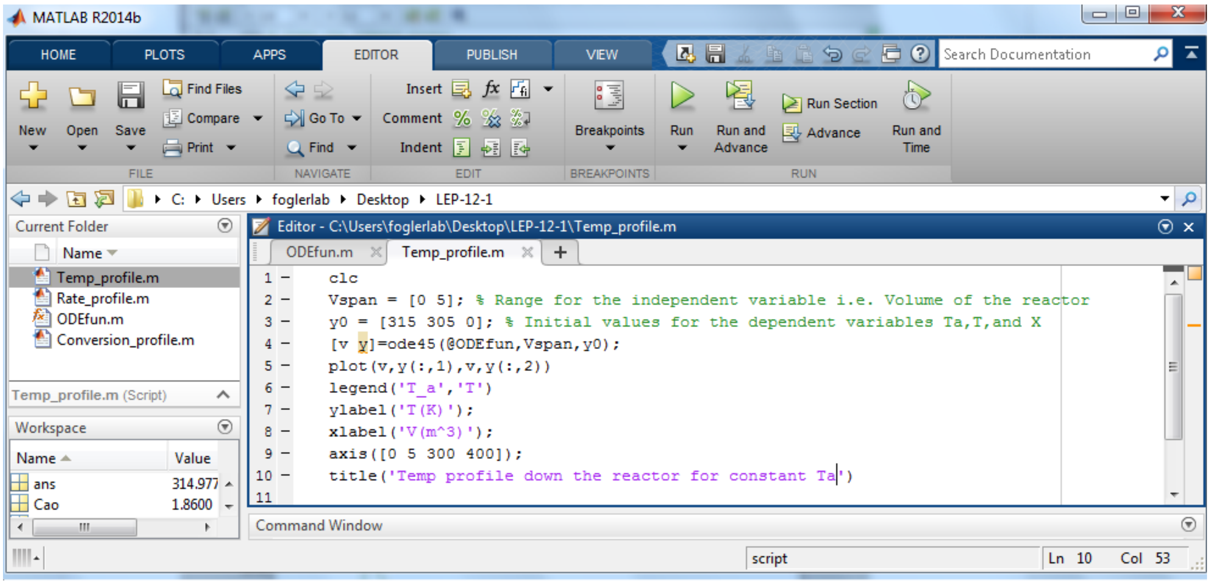


In the script file all the parameters and equation will remain same as for co-current case except the title of the graph

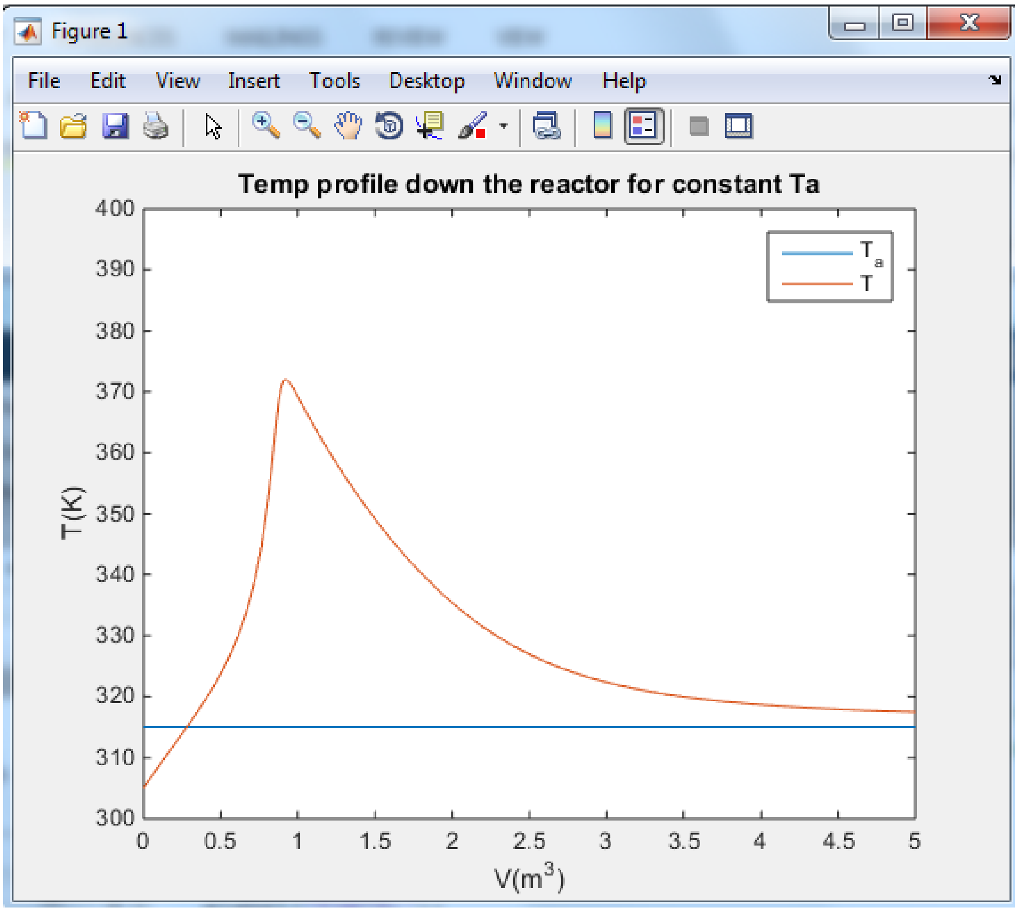
Change the title of the different graphs in all the script file and run the program to generate output.

a) Temperature profile

Step 34: The script file for Temp_profile should look like this

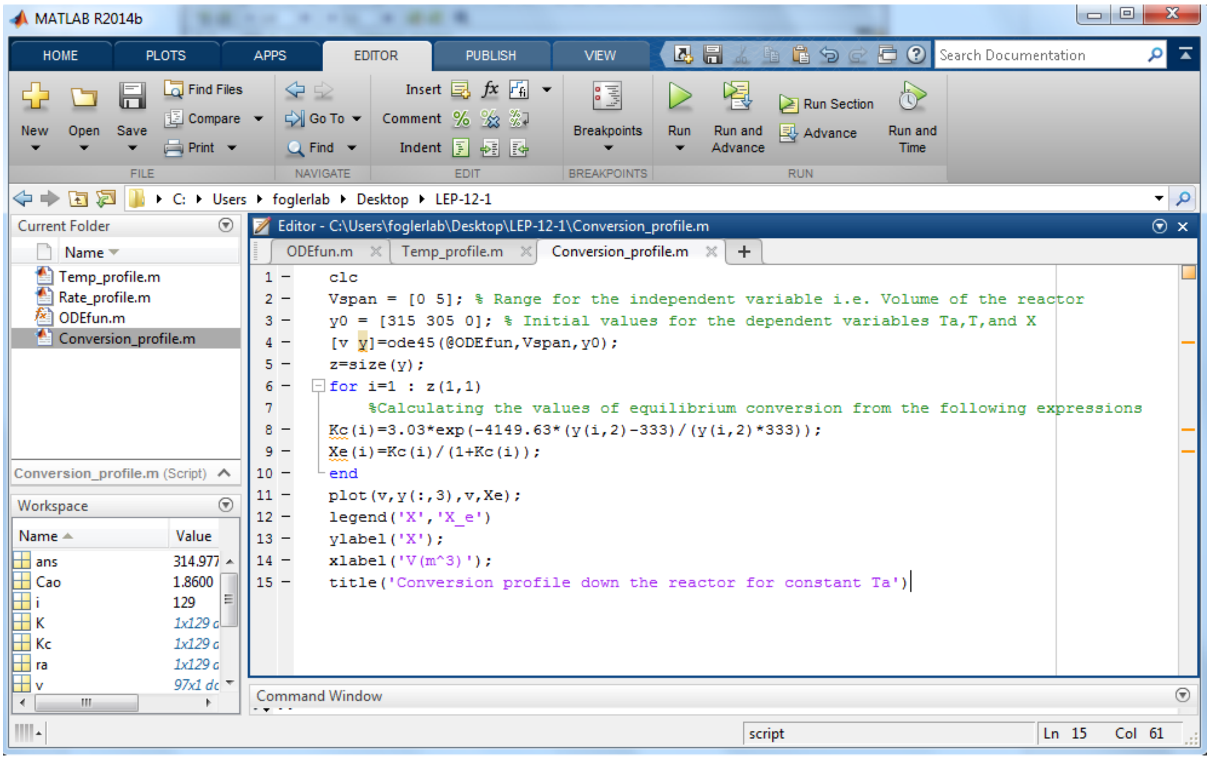


When you run the program, you should see an output like this

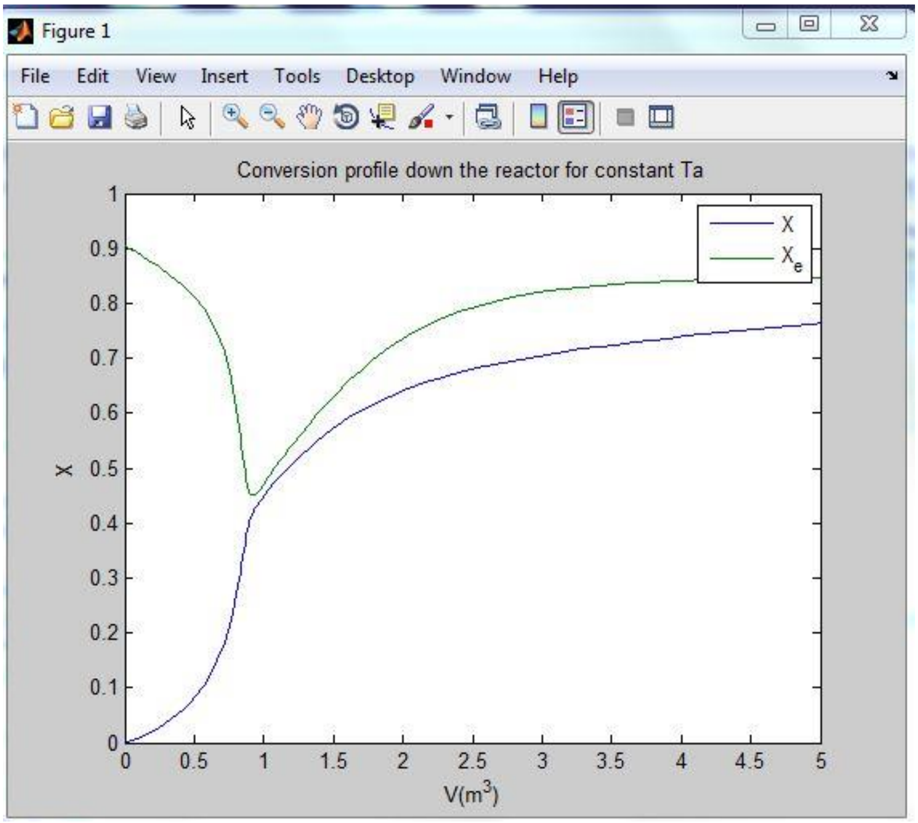


b) Conversion profile

Step 35: The script file for Conversion_profile should look like this

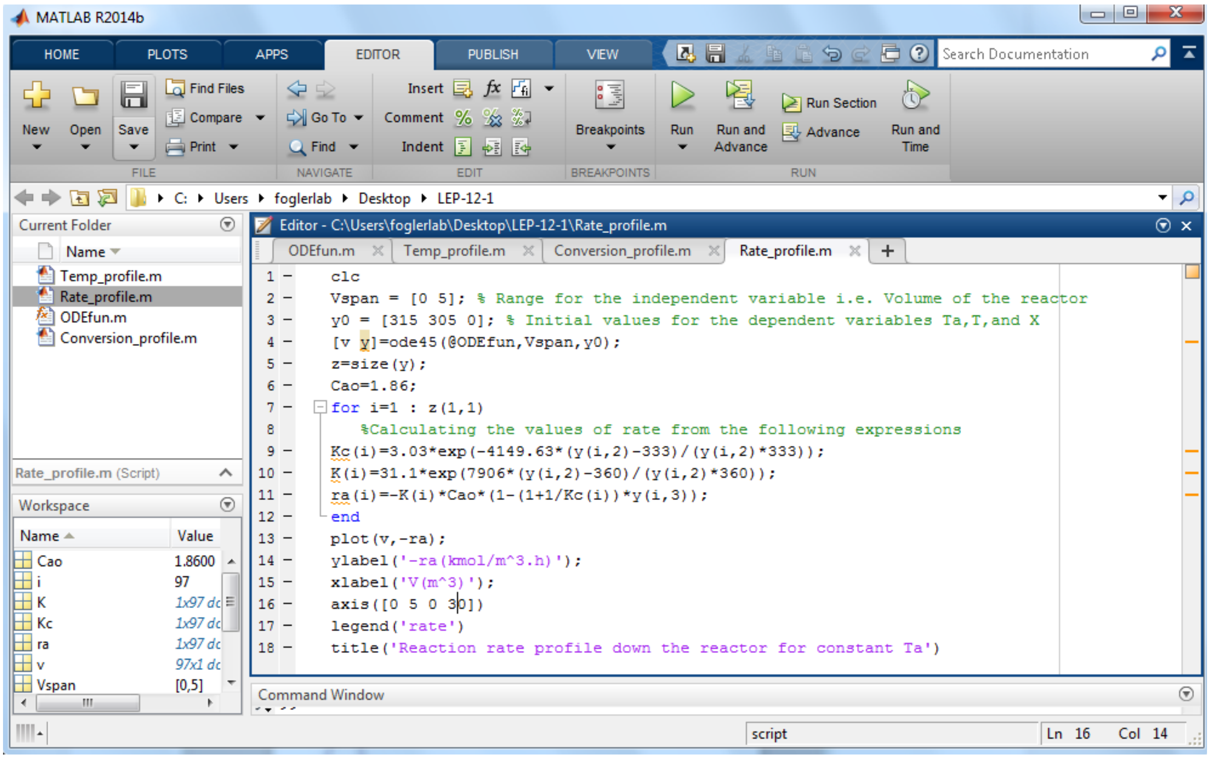


When you run this file, you should get an output like this

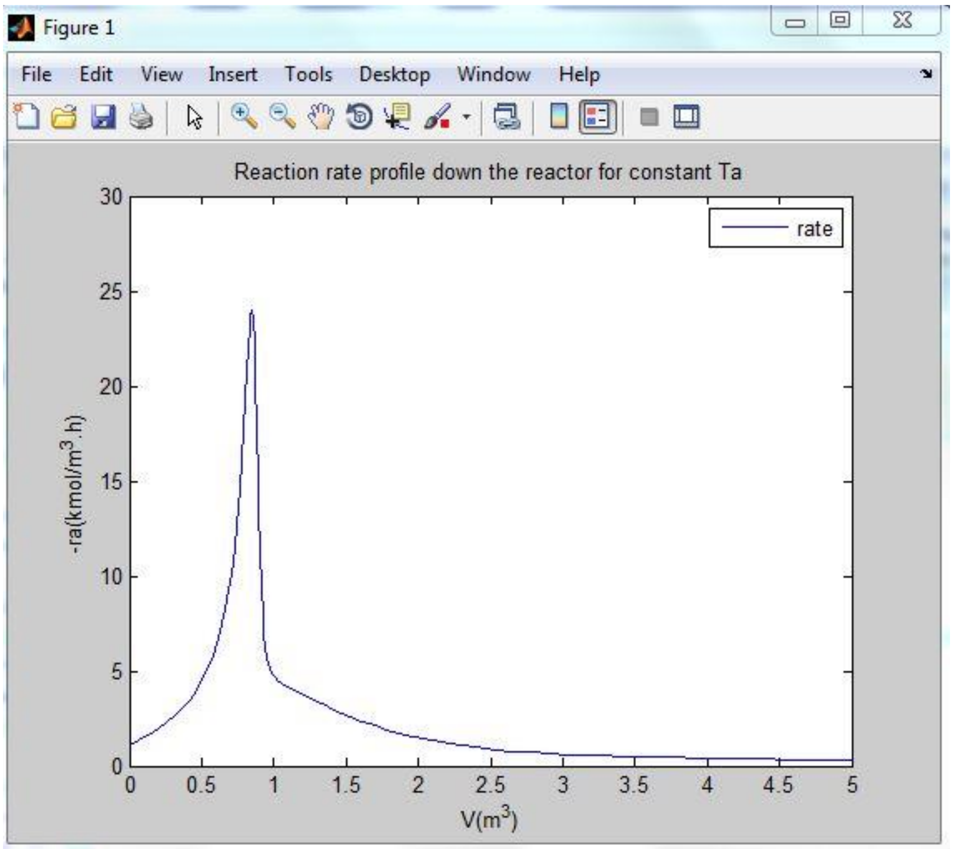


c) Rate profile

Step 36: The script file should look like this



When you run the above file, the output generated would be

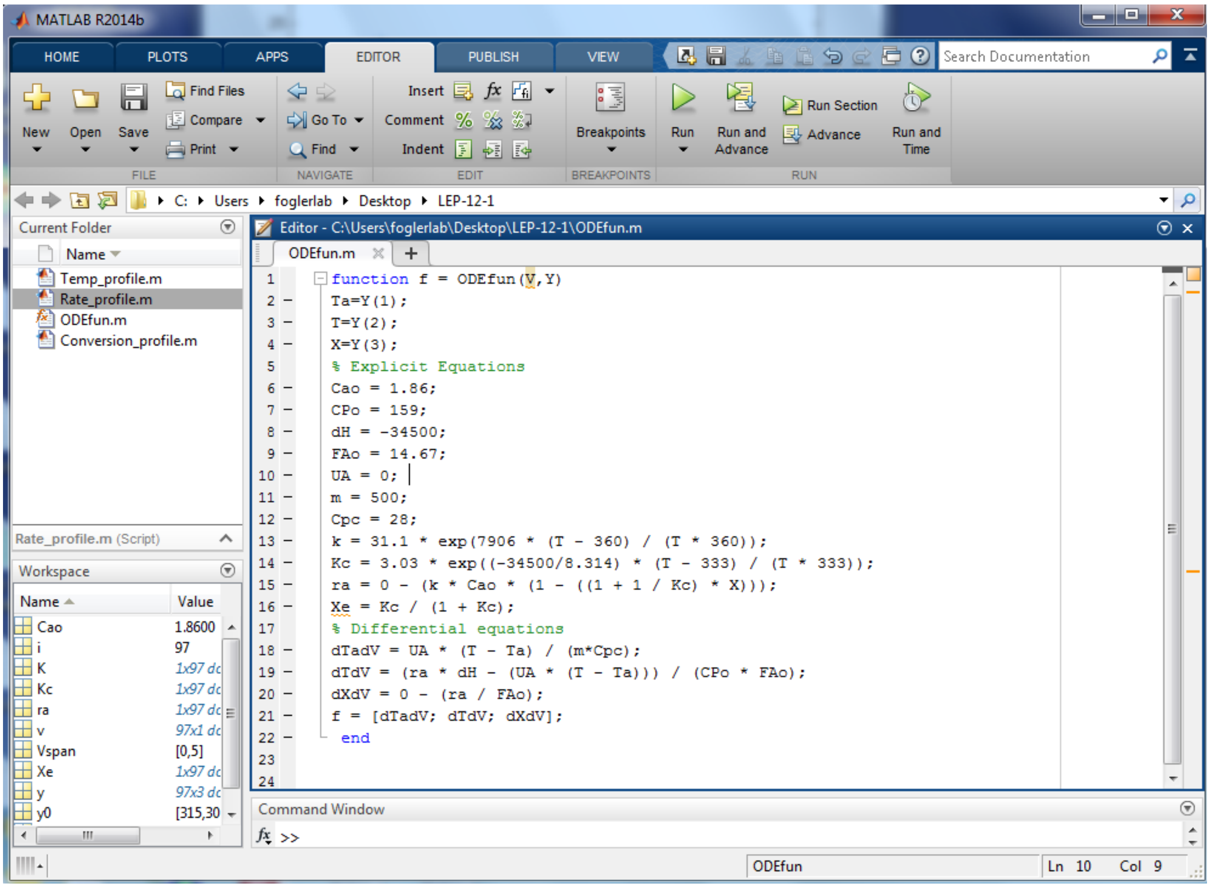


Adiabatic operation

Step 37: For the adiabatic operation, heat exchange is zero i.e. Ua=0

In the function file, modify the expression for Ua (under explicit equation section). Make Ua=0 instead of 5000

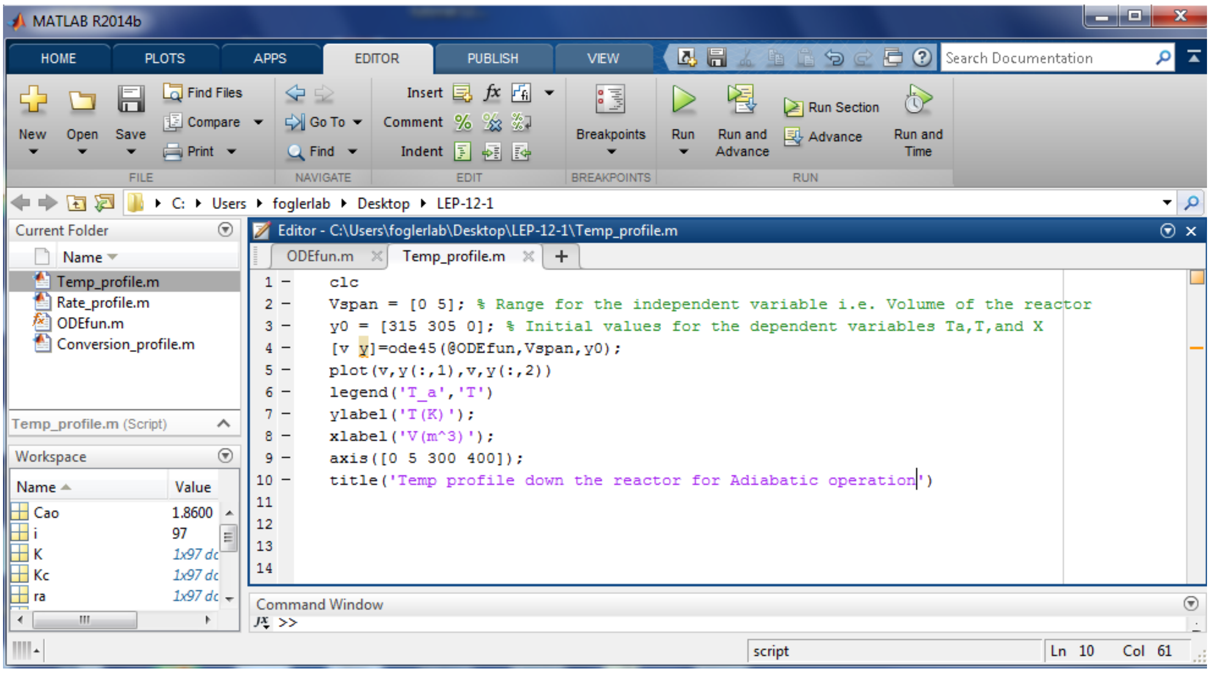
Your modified function file should look like this



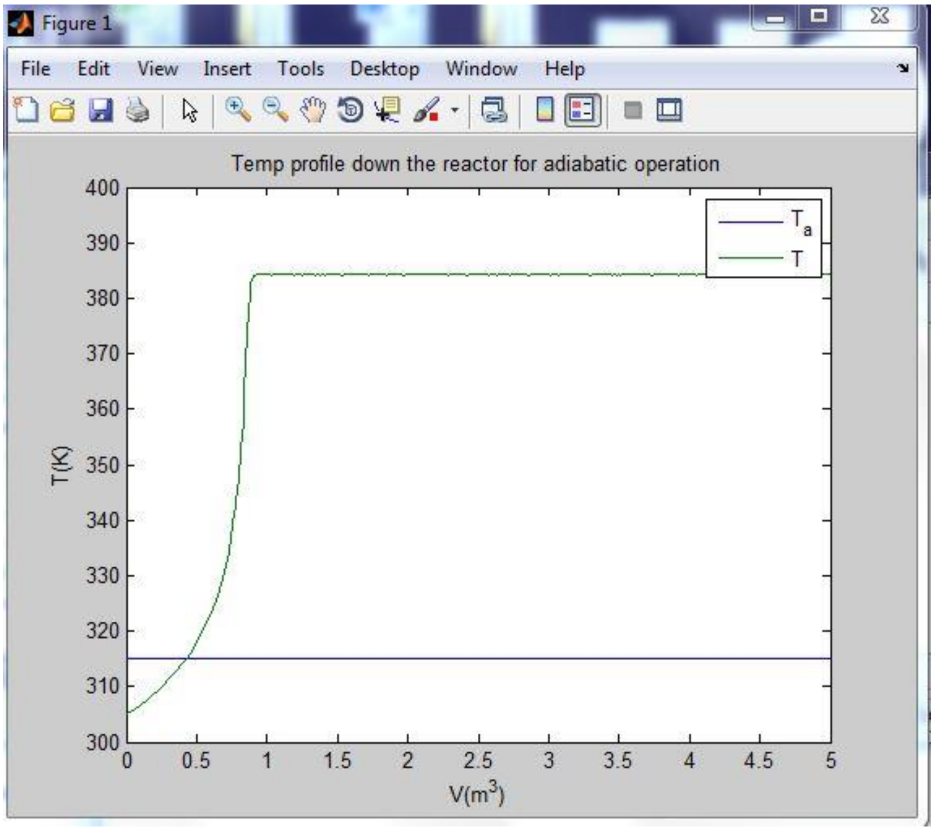
All the other expression and equation will remain same as per co-current case.
In the script file, all expression and equation will remain same as per co-current case except the graph title.

a) Temperature profile

Step 38: The script file for Temp_profile should look like this

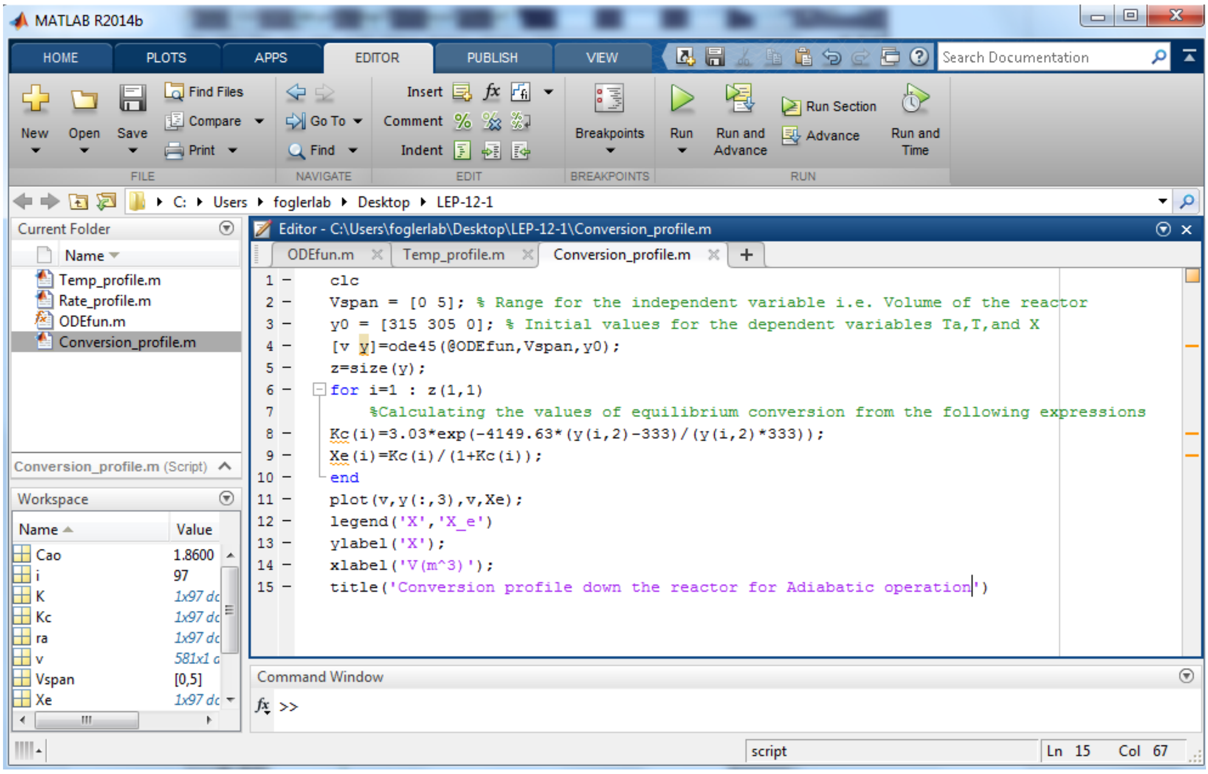


Run the above program to generate the output shown below

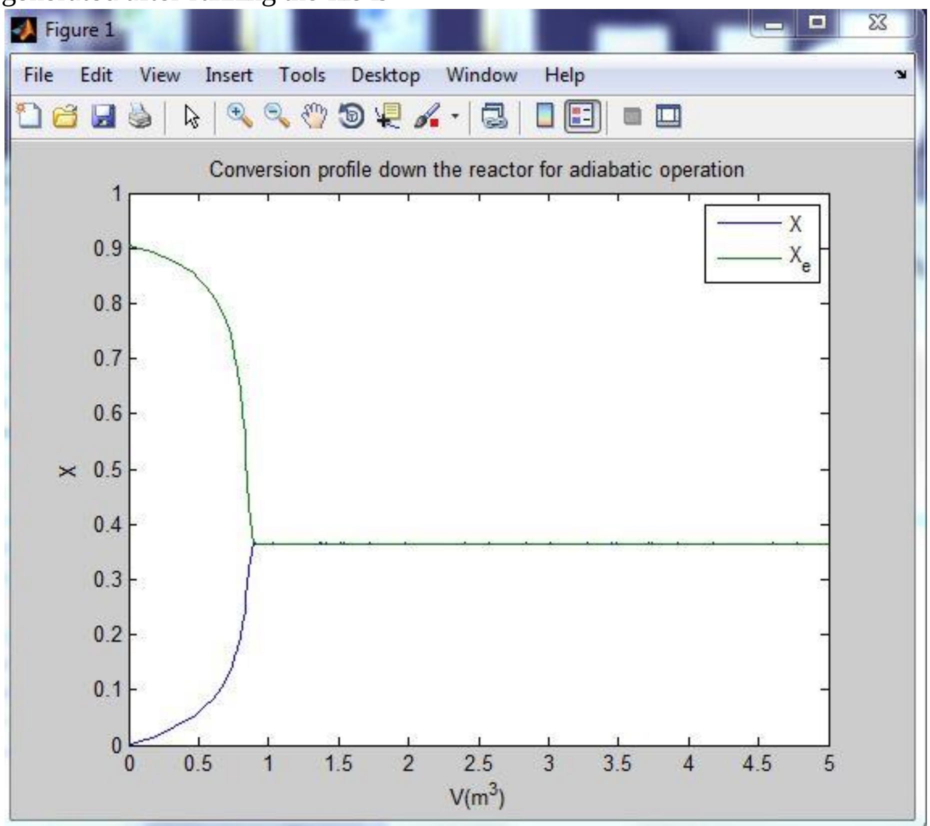


b) **<u>Conversion profile</u>**

Step 39: The script file for Conversion_profile is as shown in screenshot

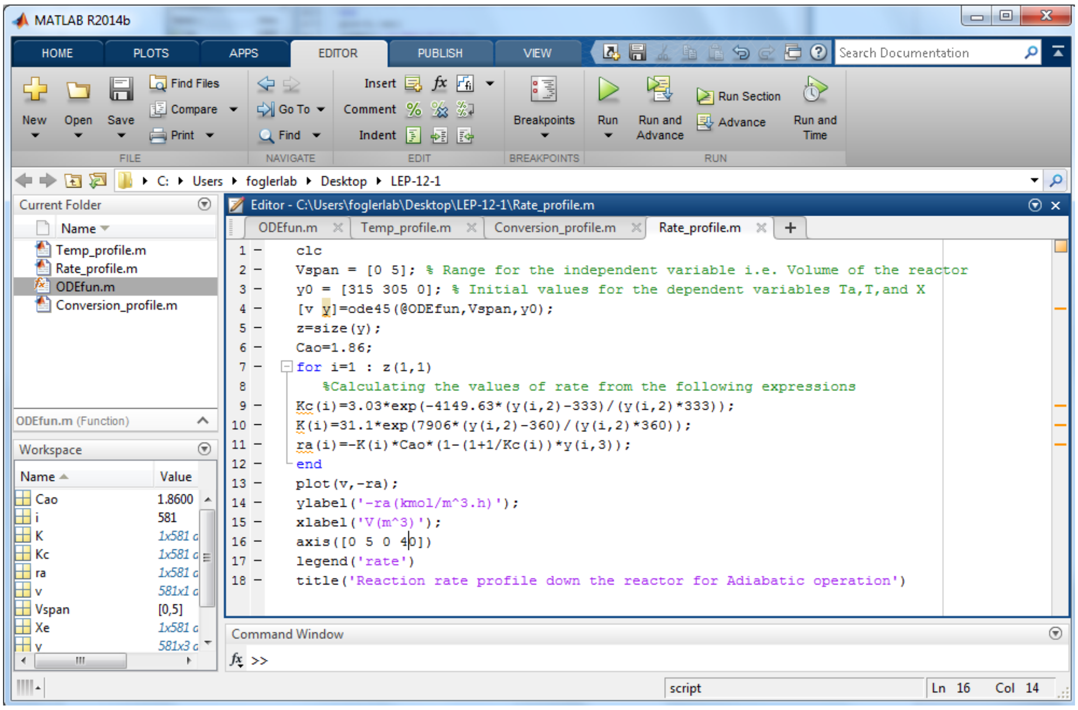


The output generated after running the file is

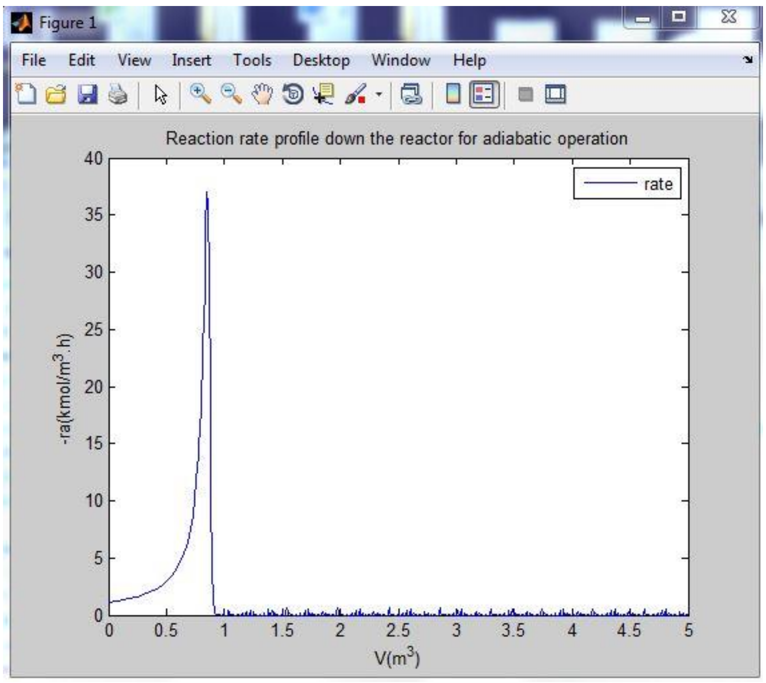


c) Rate profile

Step 40: The modified Rate_profile for adiabatic operation is



When you run the above file, the output generated is



This is all basically all you need to get started with solving differential equations in MATLAB. If you face any problem then restart MATLAB or try to solve error. To get the value of particular parameter, remove the semi-colon from the file and you will find that value is displayed on the command window