**ENR 261 Spring 2023 Chapter 15 16 17 Homework**

**General Instructions:**

Save your all your Matlab files for this chapter in the folder named **Ch015\_to\_17** located inside your local repository on your USB Memory Stick. When finished be sure to add, commit, and push your changes to your remote repository on GitHub.

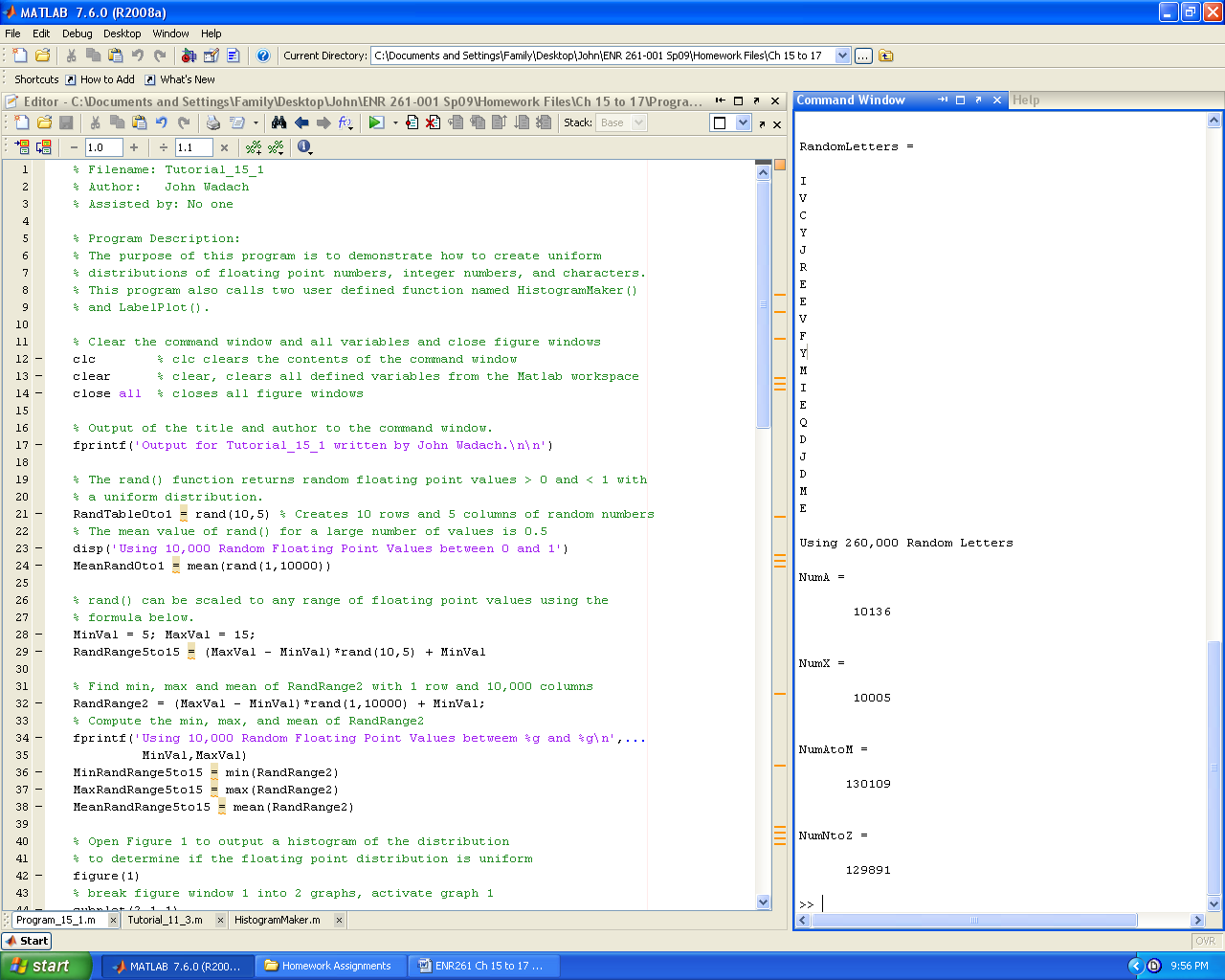
**Assigned Exercises**

1. Recreate all of the following script files and be sure to save them in your local repository on your USB memory stick, commit the changes and push them to GitHub.

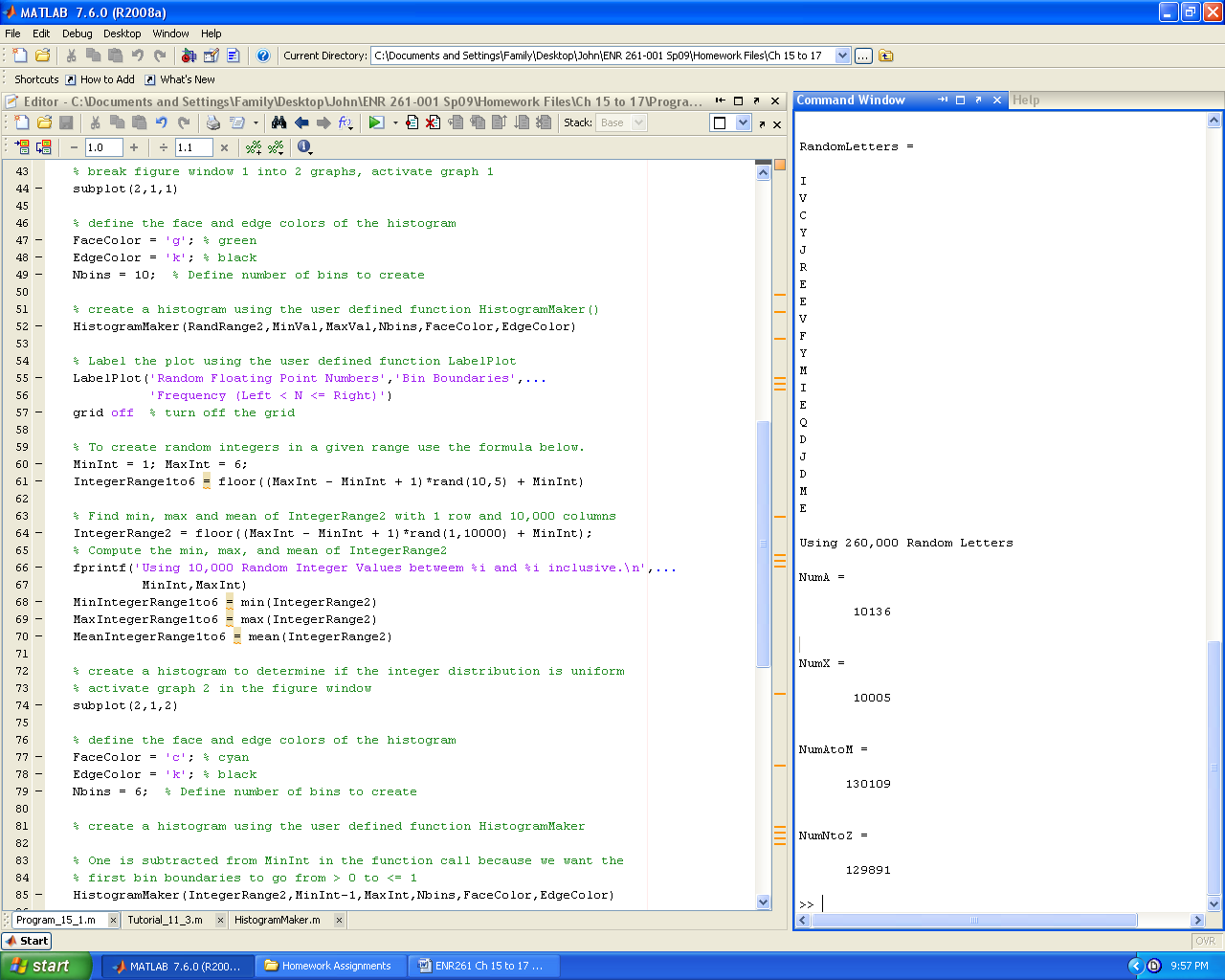
2. Use the required file names for each script file.

Required File Name: **Tutorial\_15\_1.m**

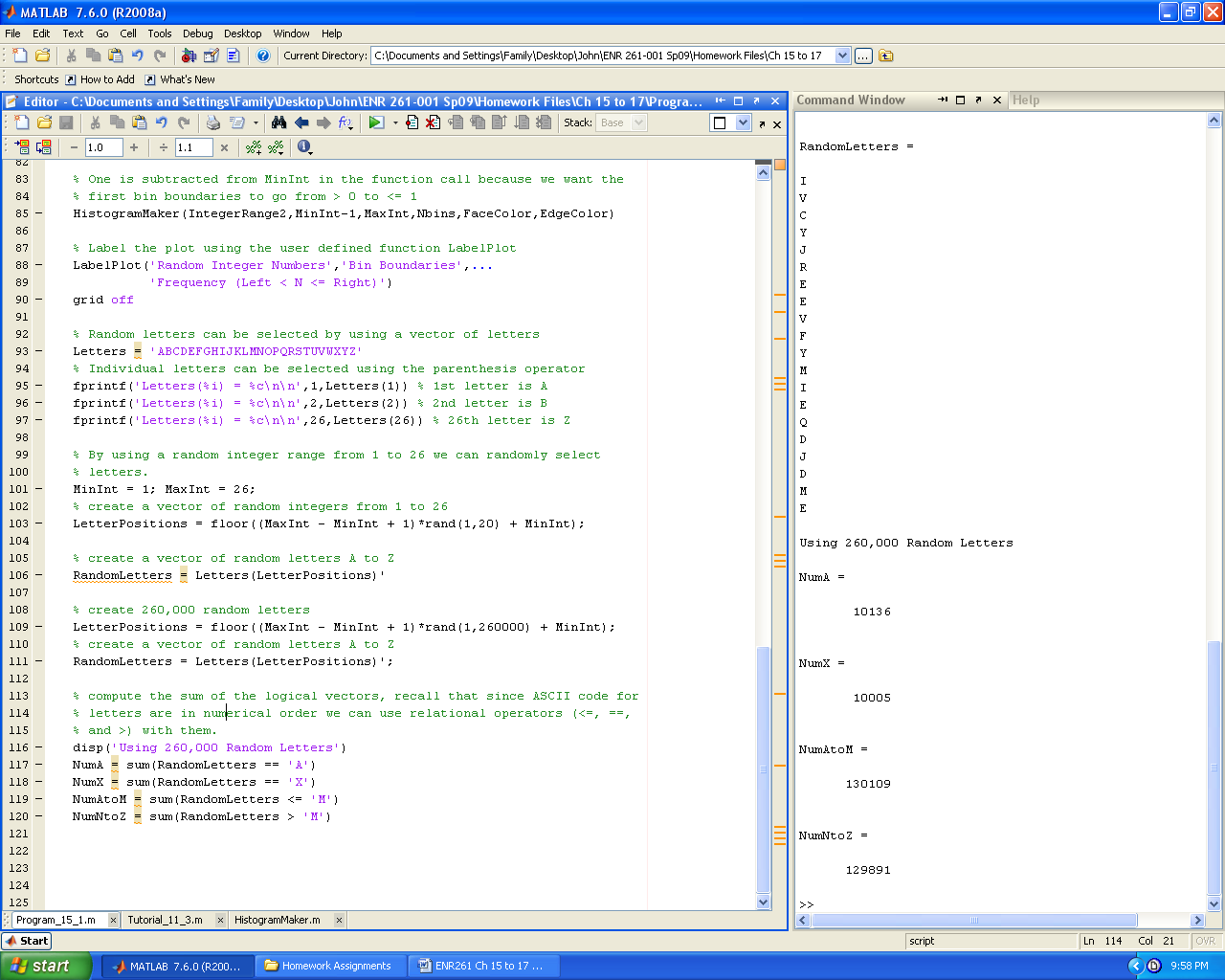
You will also have to create the user defined function named **HistogramMaker()** listed after this program.

****

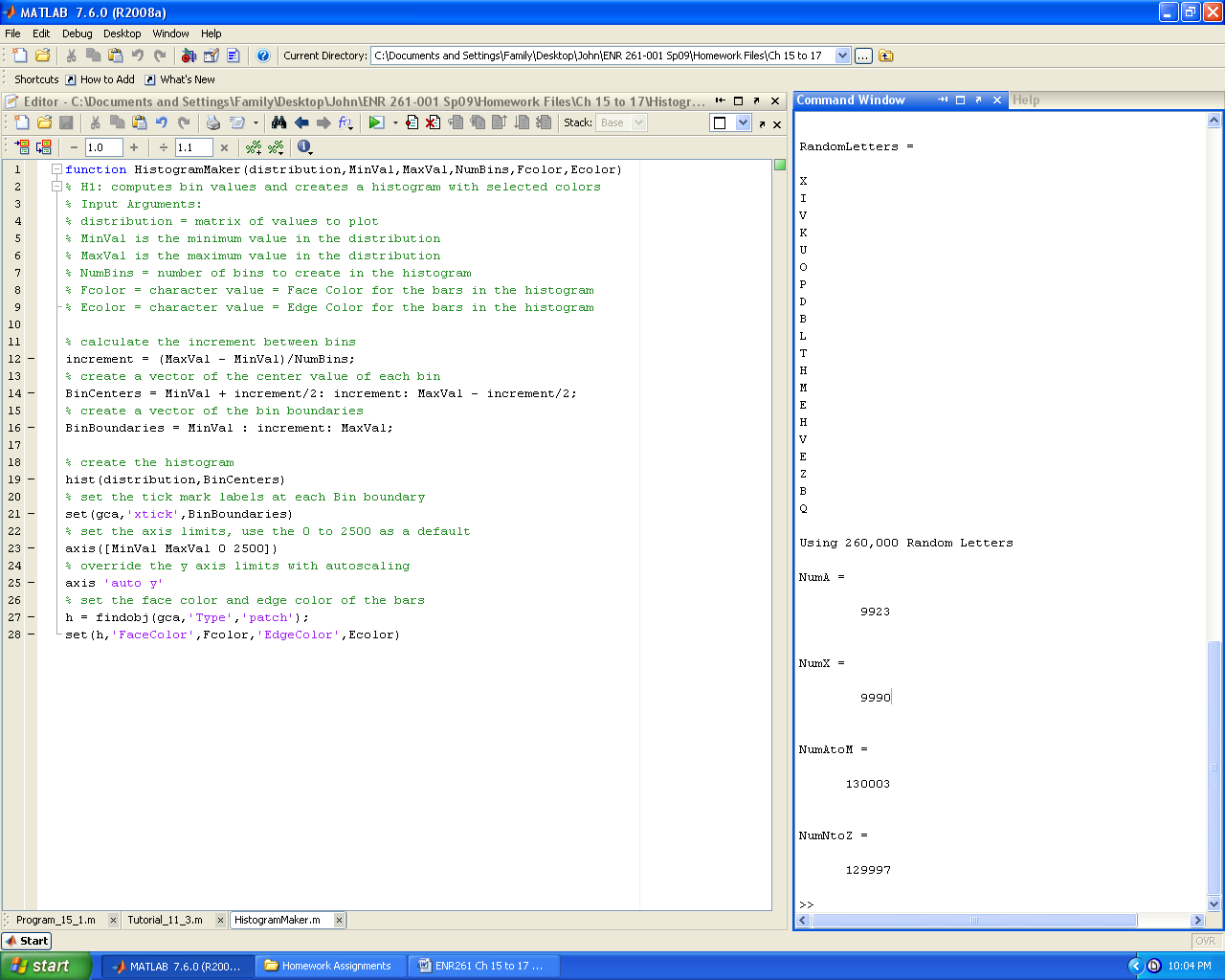
**Continued on the Next Page**

****

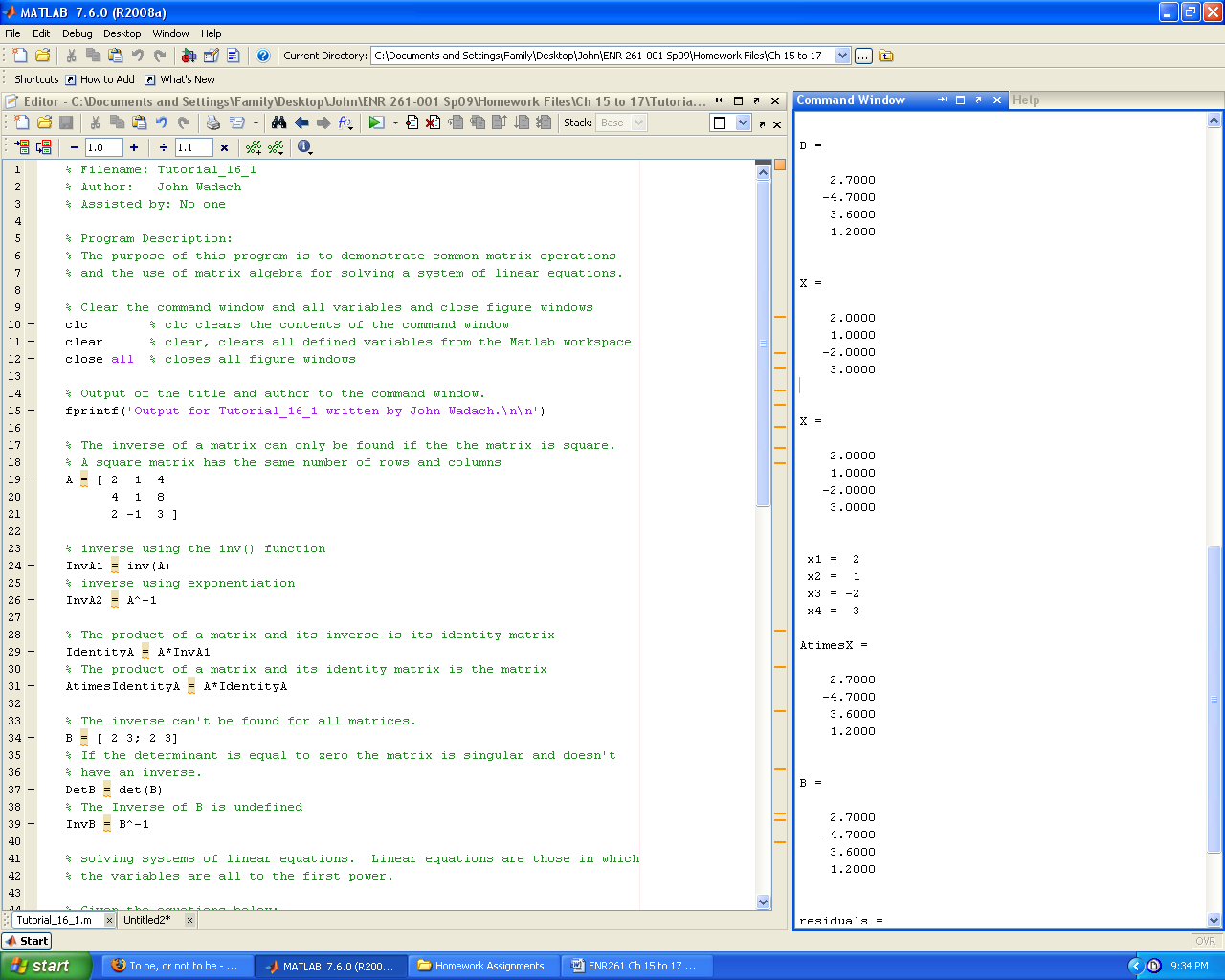
**Continued on the Next Page**

****

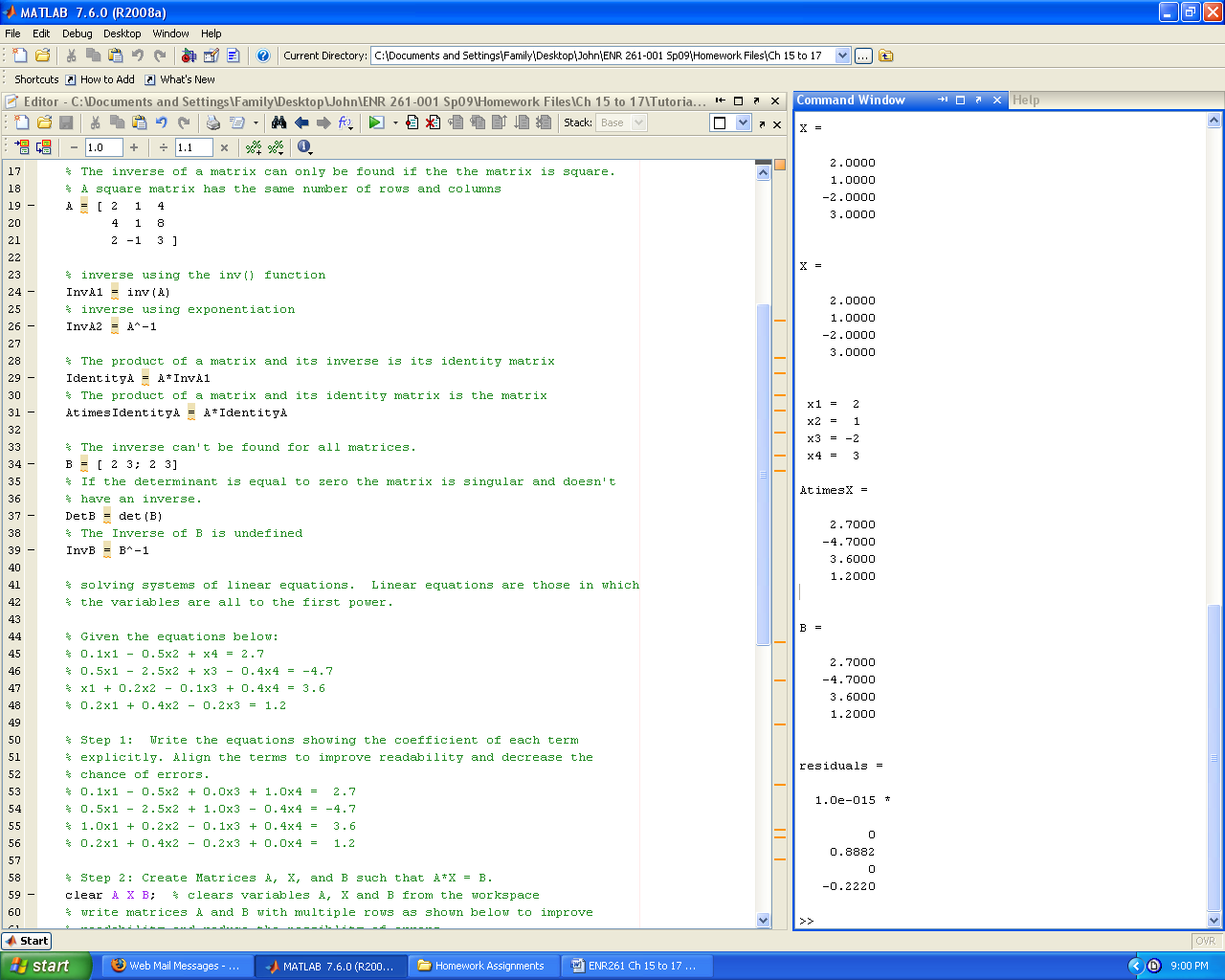
Required File Name: **HistogramMaker.m**

****

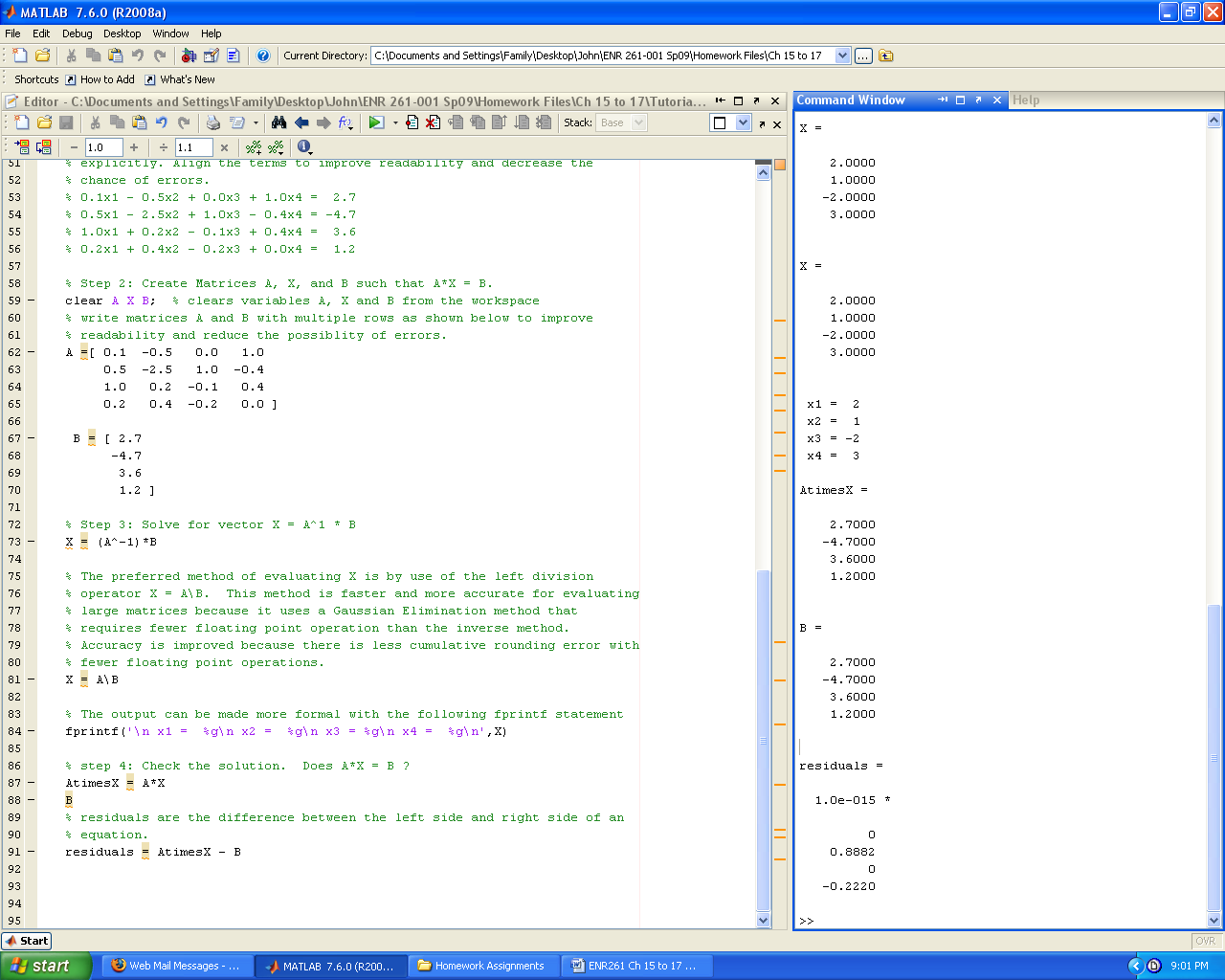
Required File Name: **Tutorial\_16\_1.m**

****

**Continued on the Next Page**

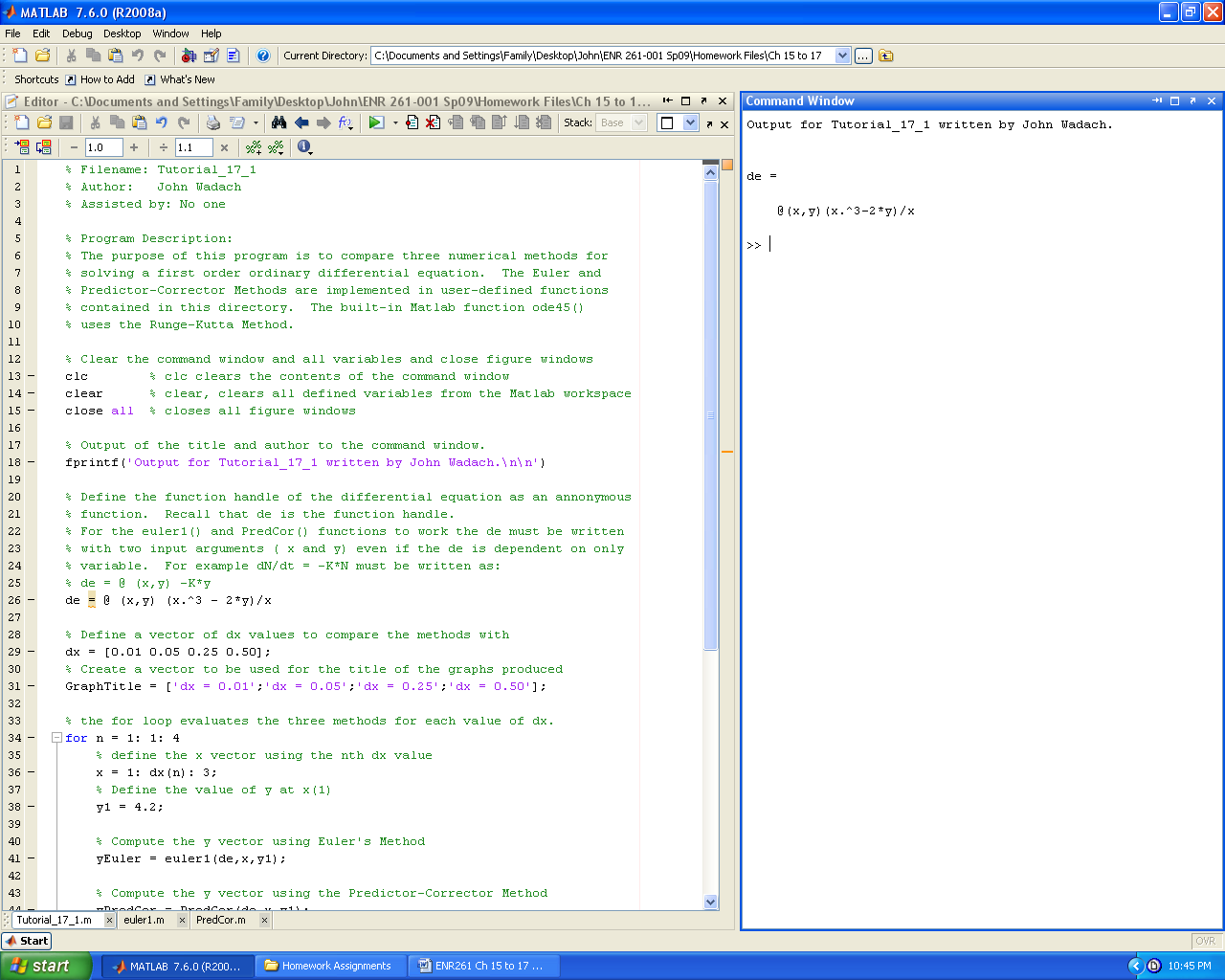
****

**Continued on the Next Page**

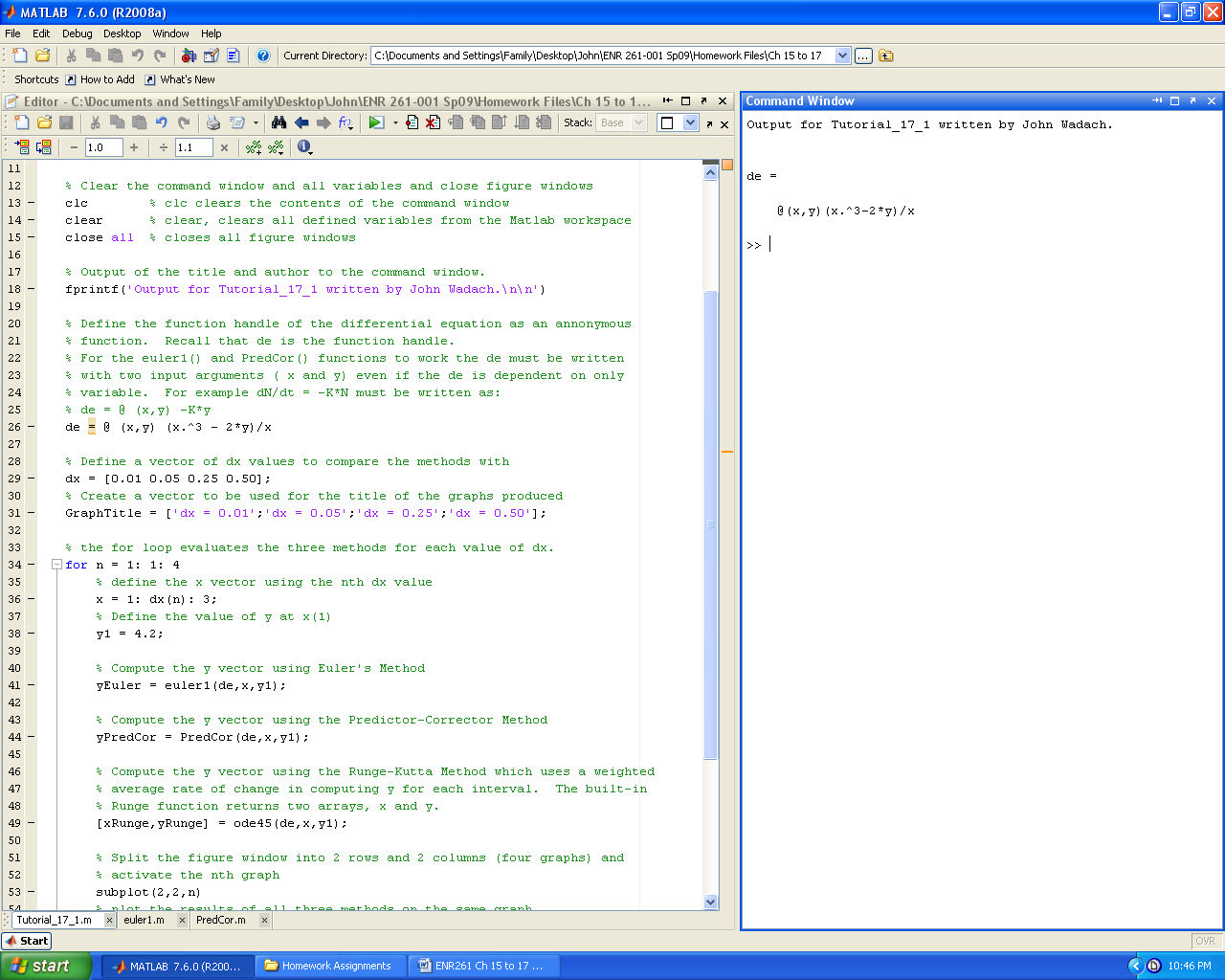
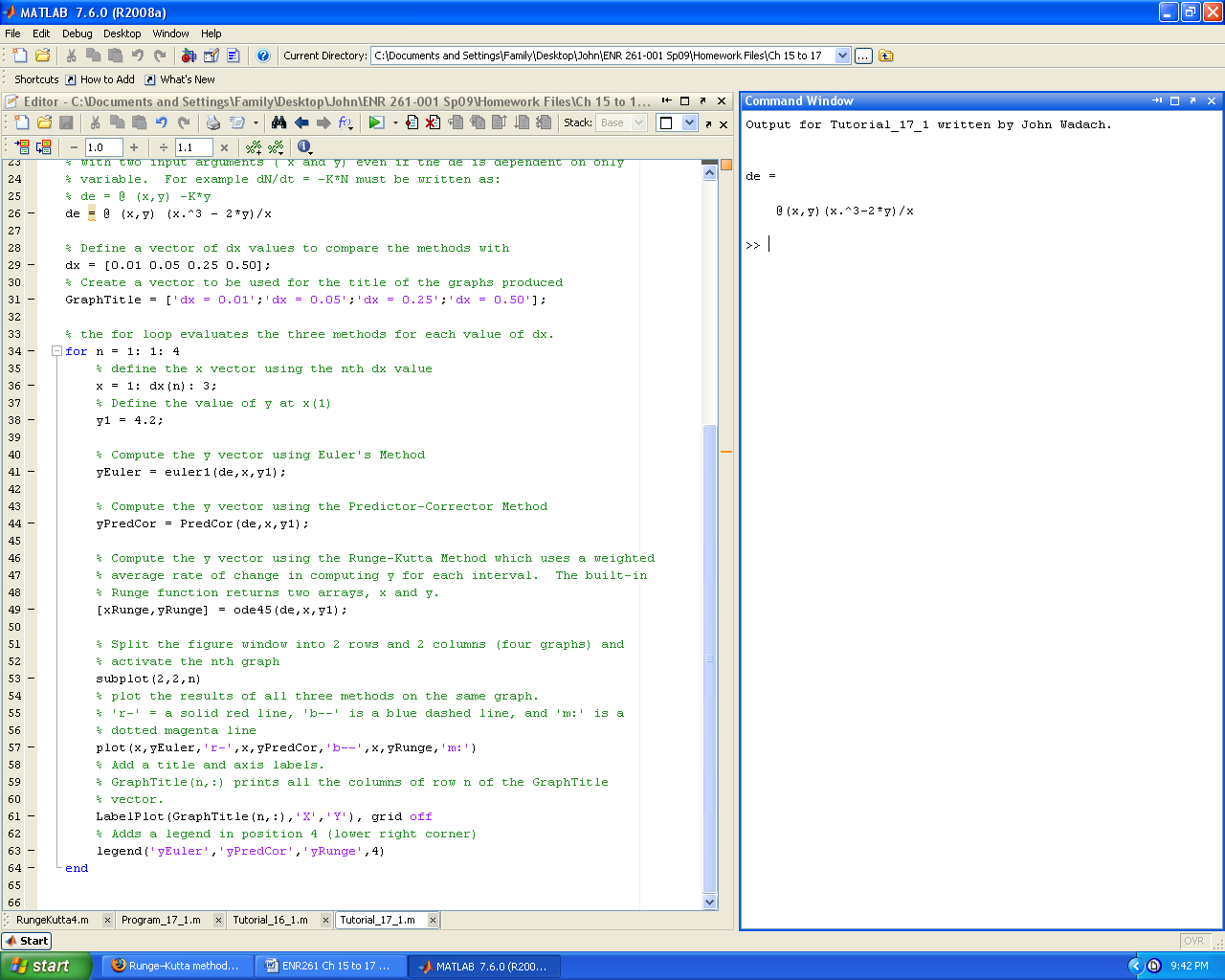
****

Required File Name: **Tutorial\_17\_1.m**

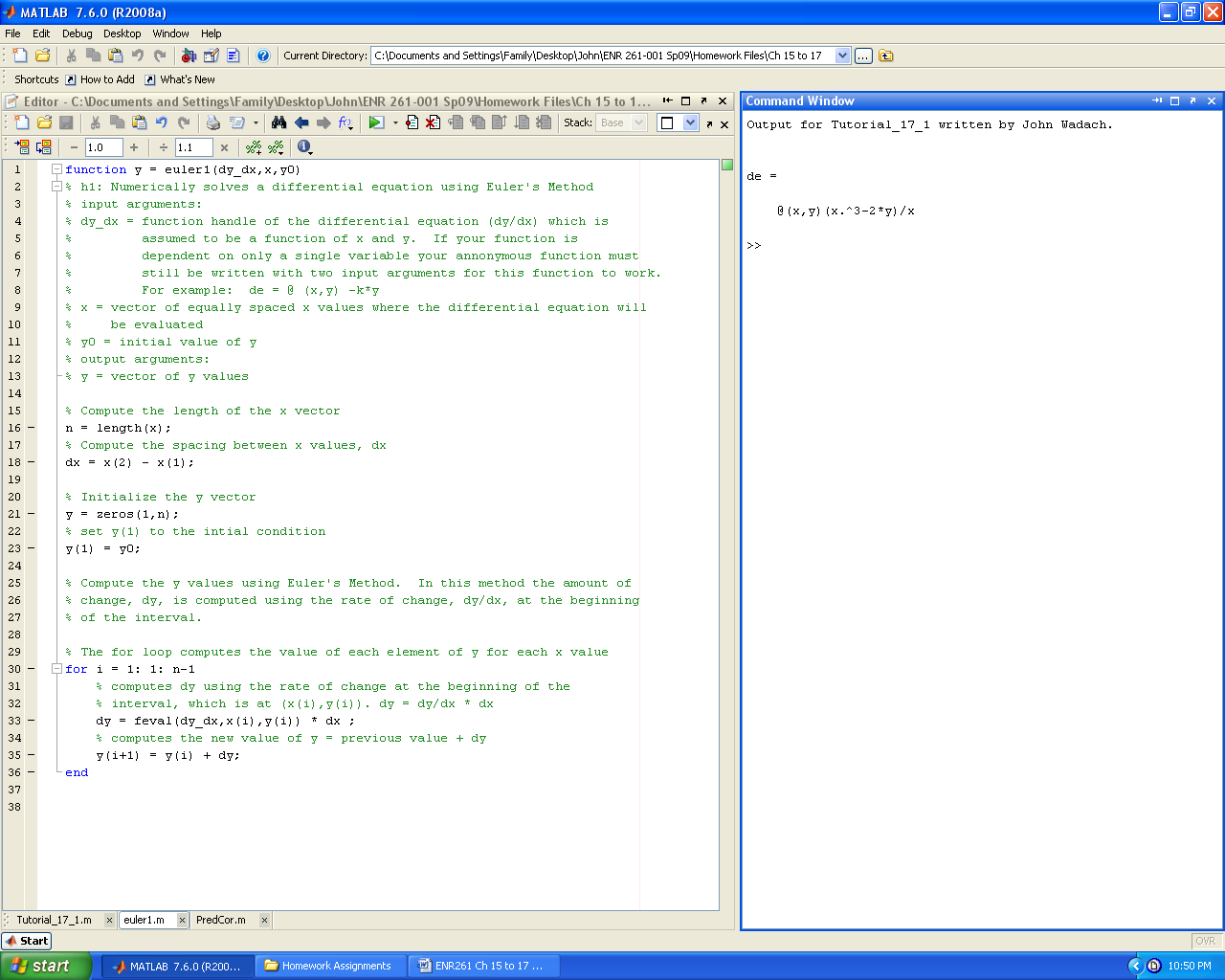
You will also have to create the user defined functions named **euler1()** and **PredCor()** listed after this program.



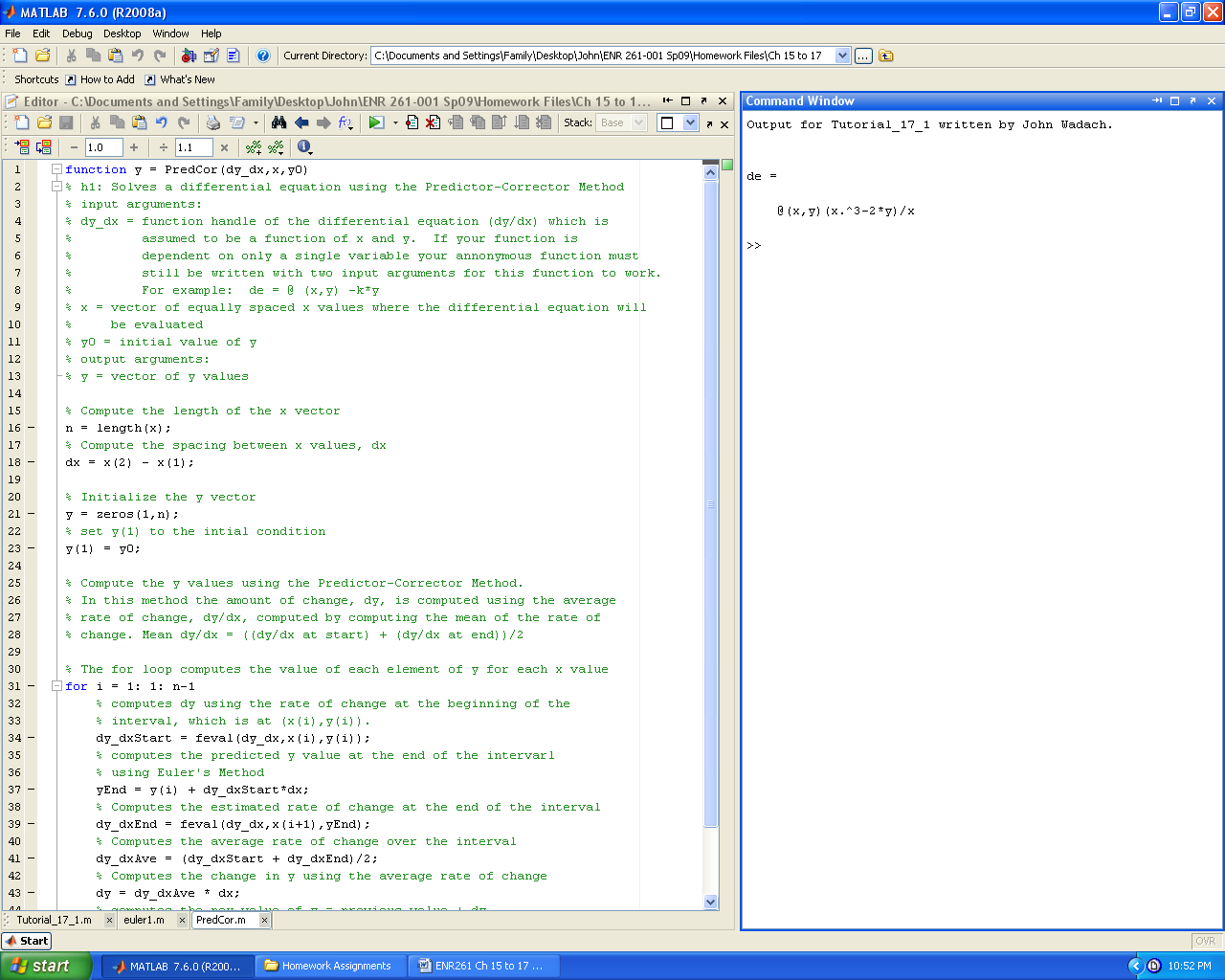
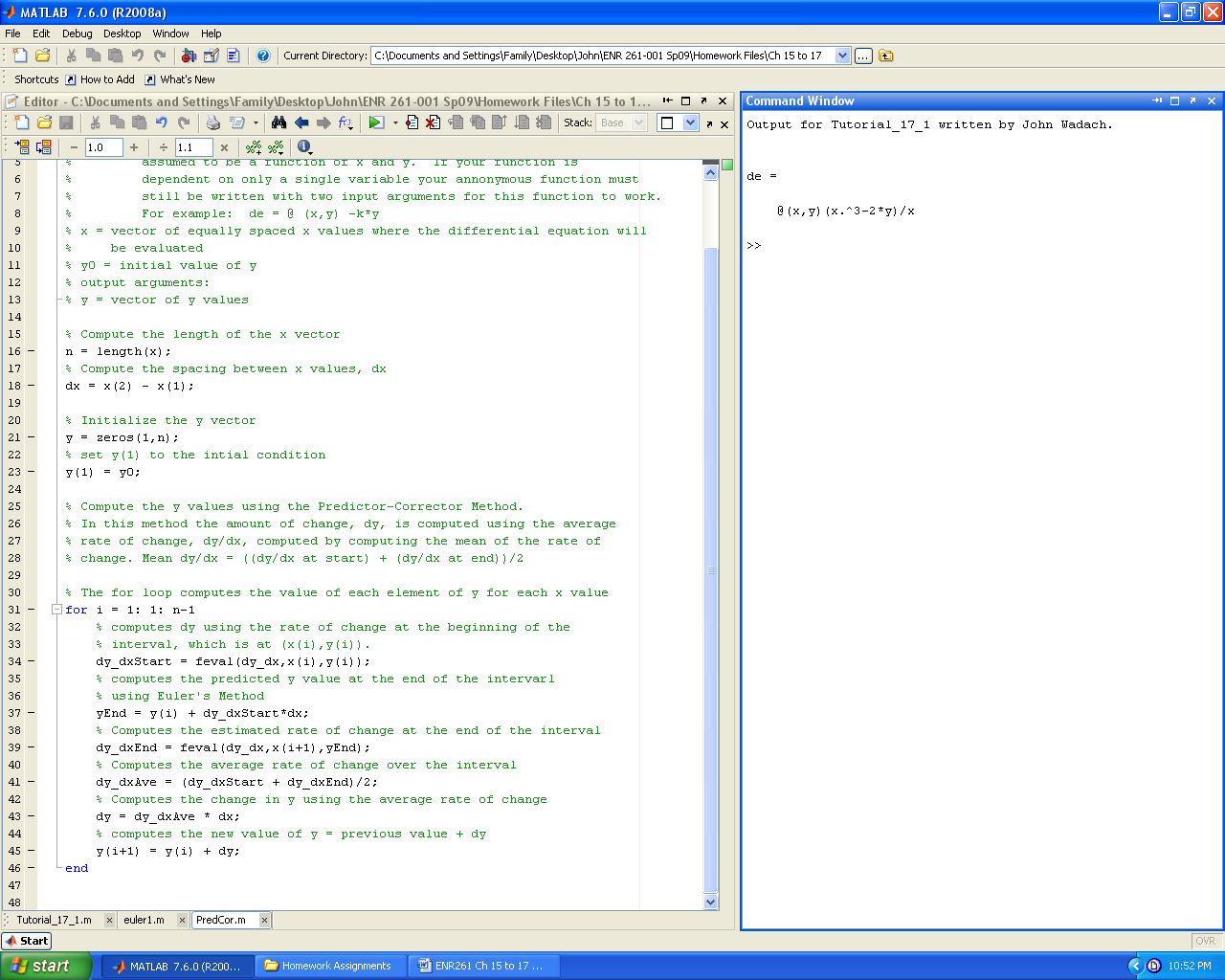
**Continued on the Next Page**

Required File Name: **euler1.m**



Required File Name: **PredCor.m**

** **

Required File Name: **Program\_15\_1.m**

The infinite monkey theorem states that a [monkey](http://en.wikipedia.org/wiki/Monkey) hitting keys at [random](http://en.wikipedia.org/wiki/Random) on a [typewriter keyboard](http://en.wikipedia.org/wiki/Typewriter_keyboard) for an [infinite](http://en.wikipedia.org/wiki/Infinite) amount of time will almost surely type the complete works of [William Shakespeare](http://en.wikipedia.org/wiki/William_Shakespeare).

In this program you will put this theorem to the test by using the rand() function to create random sequences of letters and the underscore symbol until the desired sequence is created. While attempting to create the desired sequence you must count the number of random attempts needed to create the sequence and measure the time to create the sequence.

Our ultimate goal is to create the phrase: to\_be

Please note that it will take a few minutes on average to randomly generate this phrase.

A sample of the required output is shown on the next page. **Run your program two times and** **include the output for the two trials.**

Output for Program\_15\_1 written by John Wadach.

The attempted sequence is: t

It took 13 attempts to create the sequence: t

Elapsed time is 0.001358 seconds.

The attempted sequence is: to

It took 235 attempts to create the sequence: to

Elapsed time is 0.009133 seconds.

The attempted sequence is: to\_

It took 26653 attempts to create the sequence: to\_

Elapsed time is 0.555479 seconds.

The attempted sequence is: to\_b

It took 98377 attempts to create the sequence: to\_b

Elapsed time is 2.062023 seconds.

The attempted sequence is: to\_be

It took 1.90588e+007 attempts to create the sequence: to\_be

Elapsed time is 420.934171 seconds.

>>

Note:

You can see that it would take the monkey quite a while to just type this short phrase from Hamlet without any errors.

To be, or not to be, that is the question:  
Whether 'tis nobler in the mind to suffer  
The slings and arrows of outrageous fortune,  
Or to take arms against a sea of troubles  
And by opposing end them.

Required File Name: **RungeKutta4.m (To be done in class)**

Create a user defined function to evaluate a first order differential equation using the fourth order Runge-Kutta method outlined below. **You may not** **use any built-in Matlab functions for Runge-Kutta such as ode45.**  The purpose of this exercise is for you to learn how the Runge-Kutta Method works.

## The common fourth-order Runge–Kutta method Source: Wikipedia

One member of the family of Runge–Kutta methods is so commonly used that it is often referred to as "**RK4**" or simply as "***the* Runge–Kutta method**".

Let an [initial value problem](http://en.wikipedia.org/wiki/Initial_value_problem) be specified as follows.

 y' = f(t, y), \quad y(t_0) = y_0. 

Then, the RK4 method for this problem is given by the following equations:

\begin{align}
y_{n+1} &= y_n + \tfrac{1}{6}h\left(k_1 + 2k_2 + 2k_3 + k_4 \right) \\
t_{n+1} &= t_n + h \\
\end{align}

where *yn* + 1 is the RK4 approximation of *y*(*tn* + 1), and

\begin{align} 
k_1 &= f(t_n, y_n)
\\
k_2 &= f(t_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}h k_1)
\\
k_3 &= f(t_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}h k_2)
\\
k_4 &= f(t_n + h, y_n + h k_3)
\end{align}

Thus, the next value (*yn* + 1) is determined by the present value (*yn*) plus the product of the size of the interval (*h*) and an estimated [slope](http://en.wikipedia.org/wiki/Slope). The slope is a [weighted average](http://en.wikipedia.org/wiki/Weighted_average) of slopes:

* *k*1 is the slope at the beginning of the interval;
* *k*2 is the slope at the midpoint of the interval, using slope *k*1 to determine the value of *y* at the point *tn* + *h* / 2 using [Euler's method](http://en.wikipedia.org/wiki/Euler%27s_method);
* *k*3 is again the slope at the midpoint, but now using the slope *k*2 to determine the *y*-value;
* *k*4 is the slope at the end of the interval, with its *y*-value determined using *k*3.

In averaging the four slopes, greater weight is given to the slopes at the midpoint:

\mbox{slope} = \tfrac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4).

The RK4 method is a fourth-order method, meaning that the error per step is [on the order of](http://en.wikipedia.org/wiki/Big_O_notation) *h*5, while the total accumulated error has order *h*4.

Required File Name: **Program\_17\_1.m**

Modify Tutorial\_17\_1 by deleting the Matlab ode45() function and replacing it with your RungeKutta4() function. Plot the three methods as shown below.

