**ENR 261 Spring 2023 Chapter 3 Homework**

**General Instructions:**

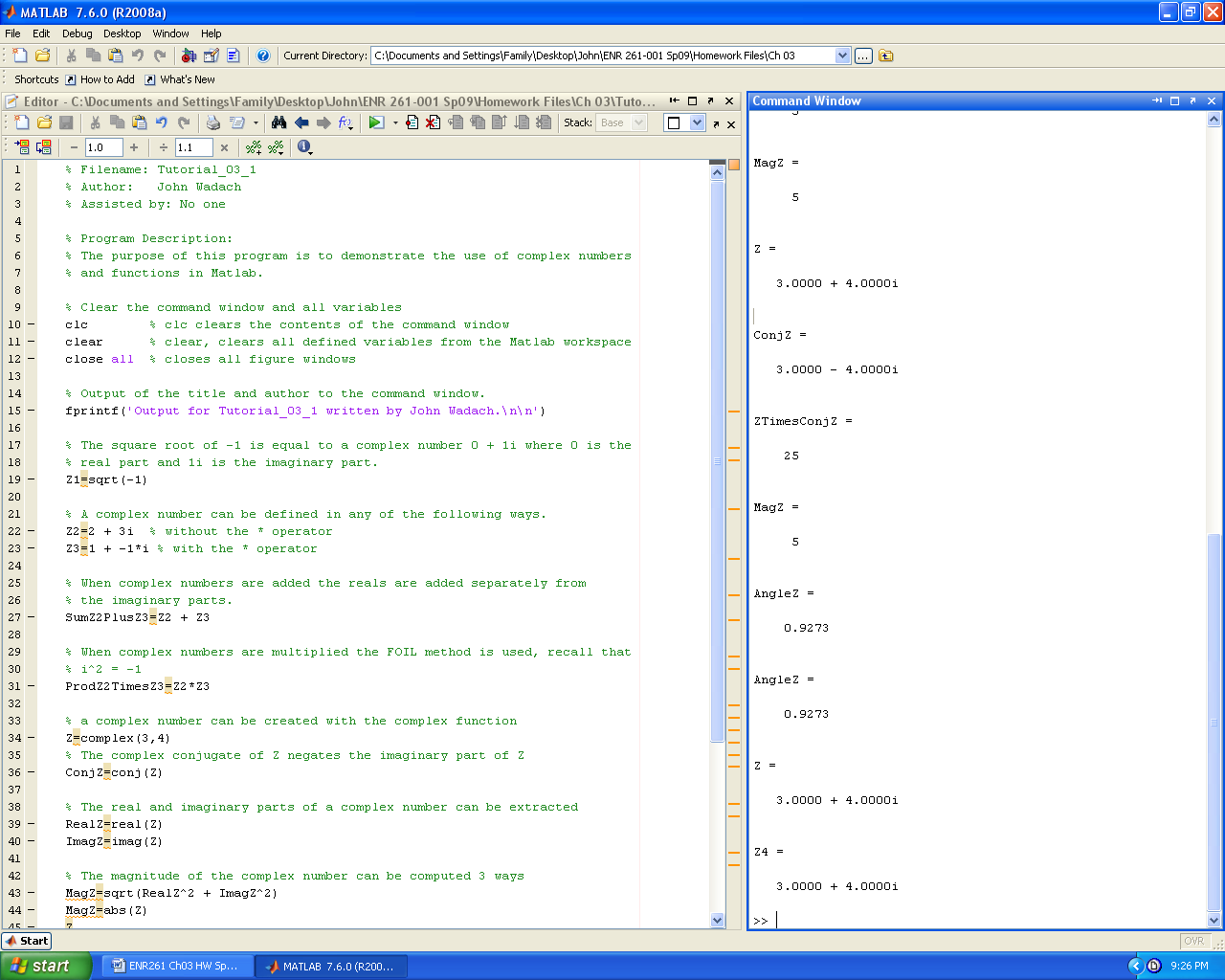
Save your all your Matlab files for this chapter in the folder named **Ch03** located inside your local repository on your USB Memory Stick. When finished be sure to add, commit, and push your changes to your remote repository on GitHub.

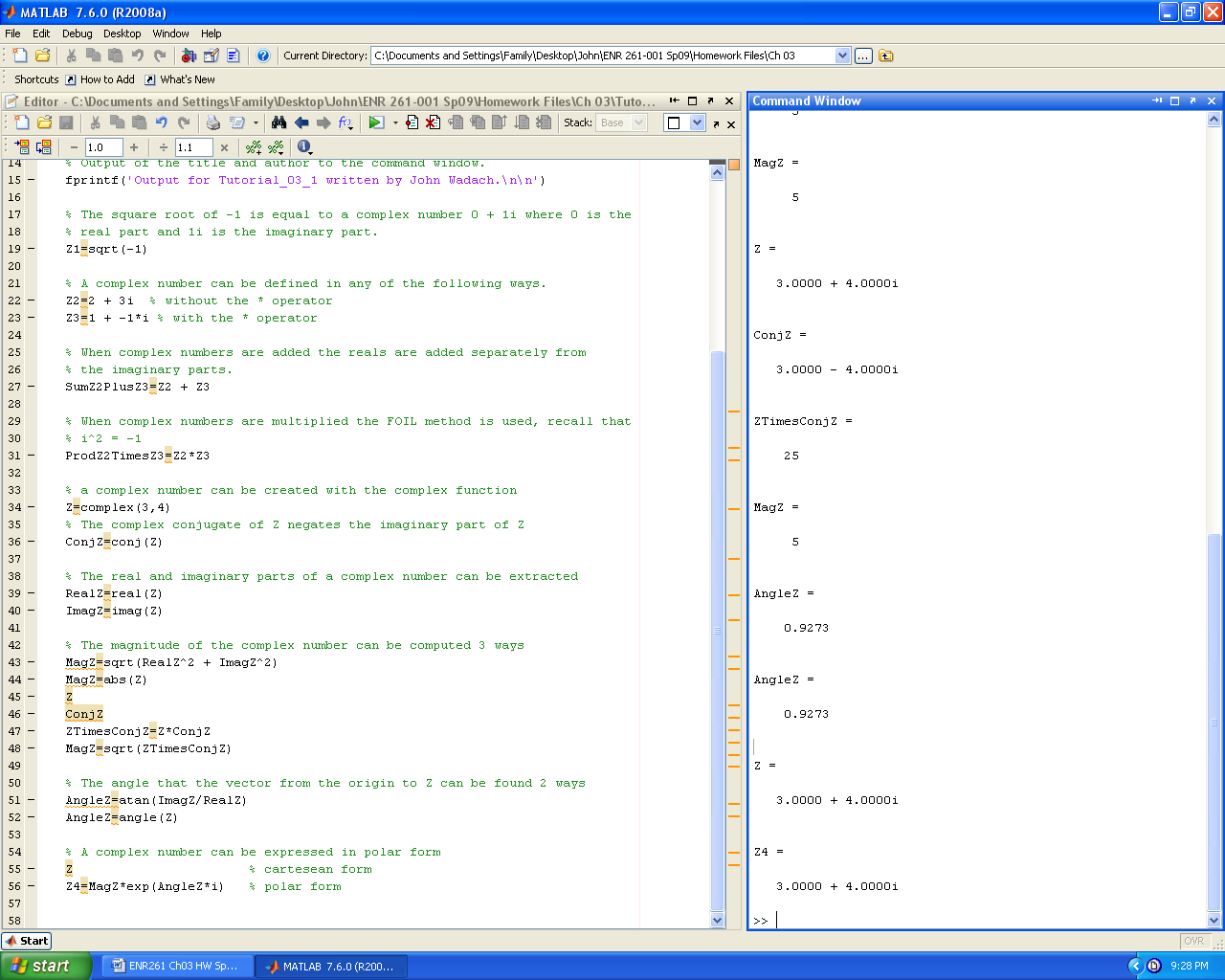
**Assigned Exercises**

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.

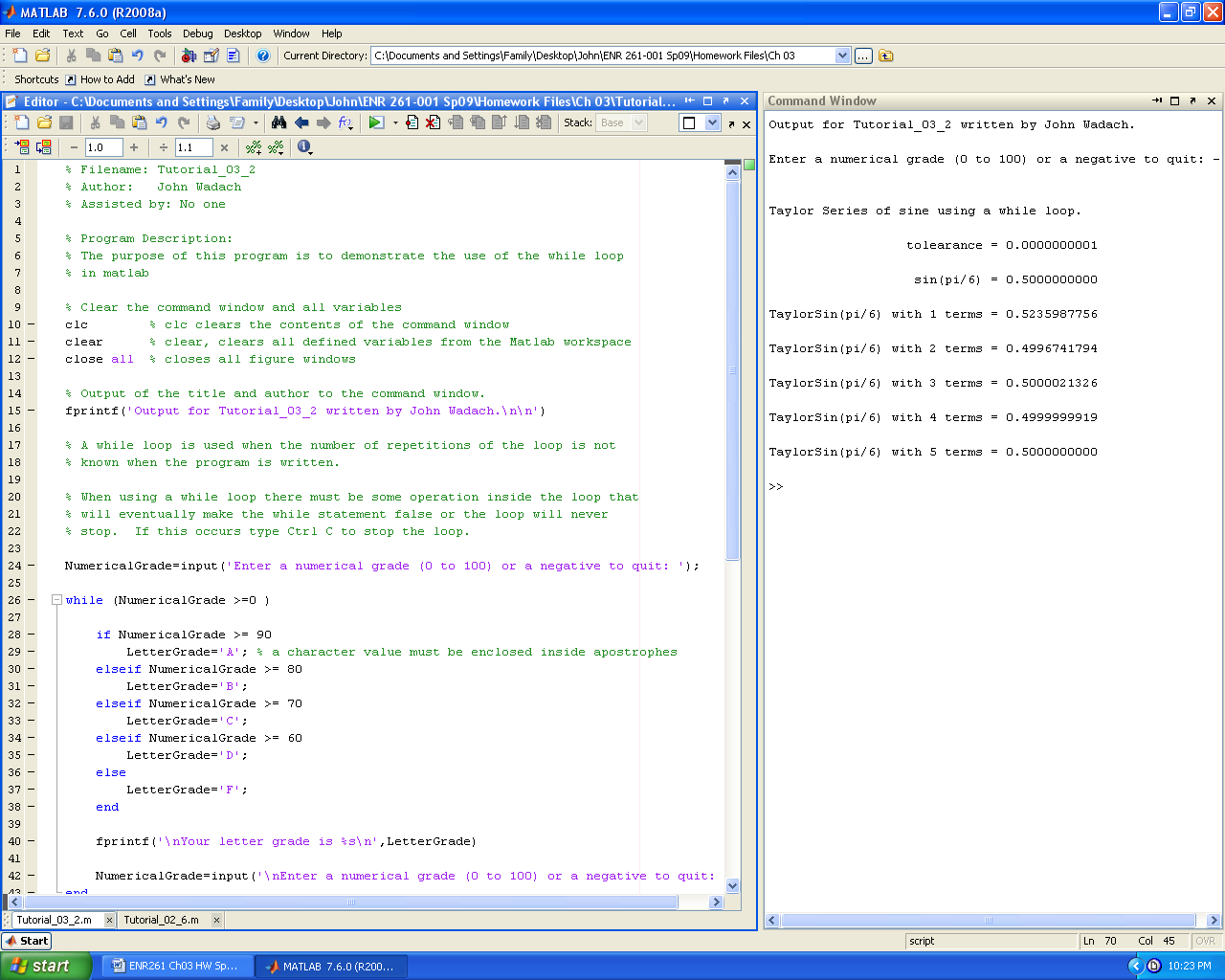
2. Use the required file names for each script file.

Required File Name: **Tutorial\_03\_1.m**

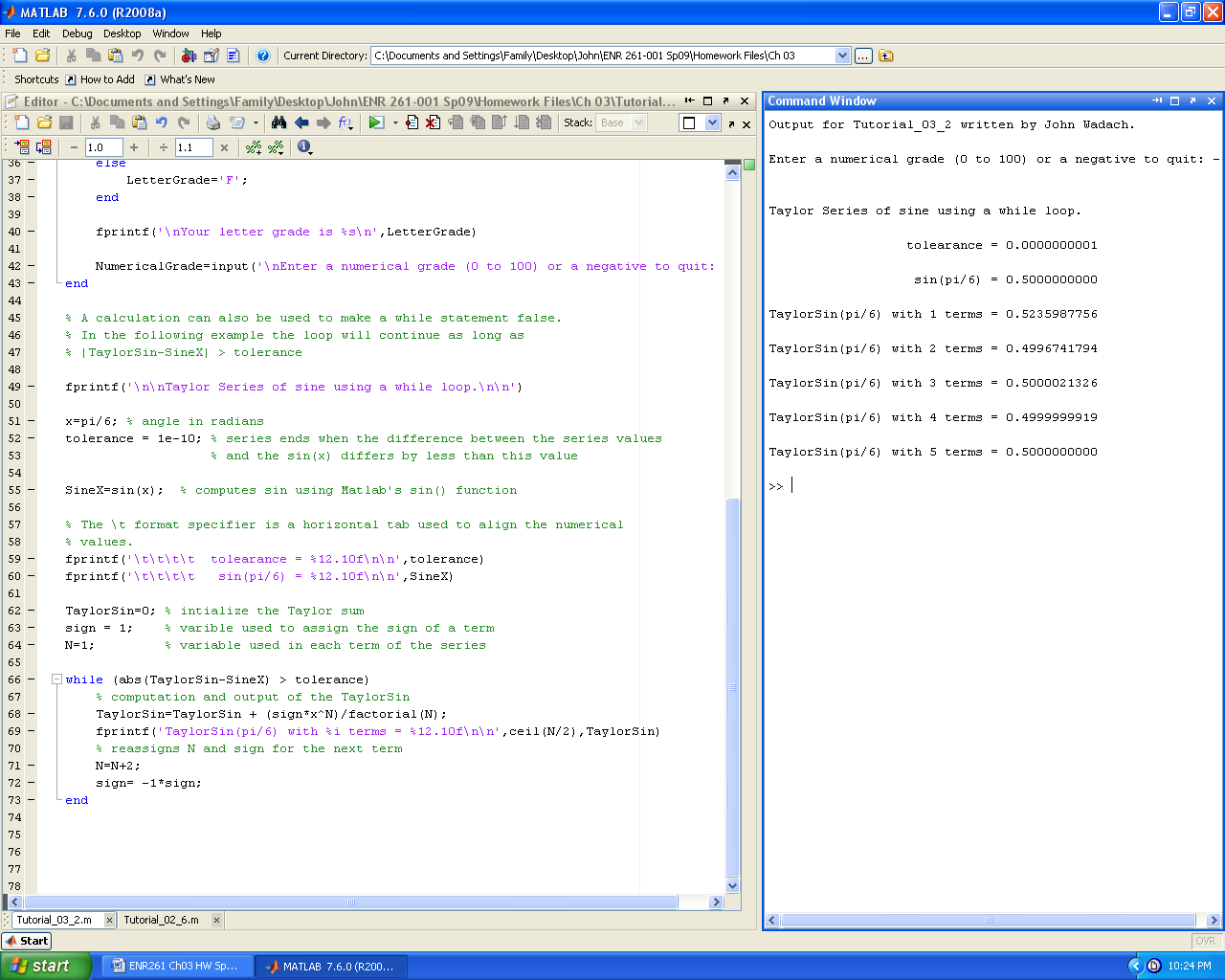
**Continued on the Next Page**

****

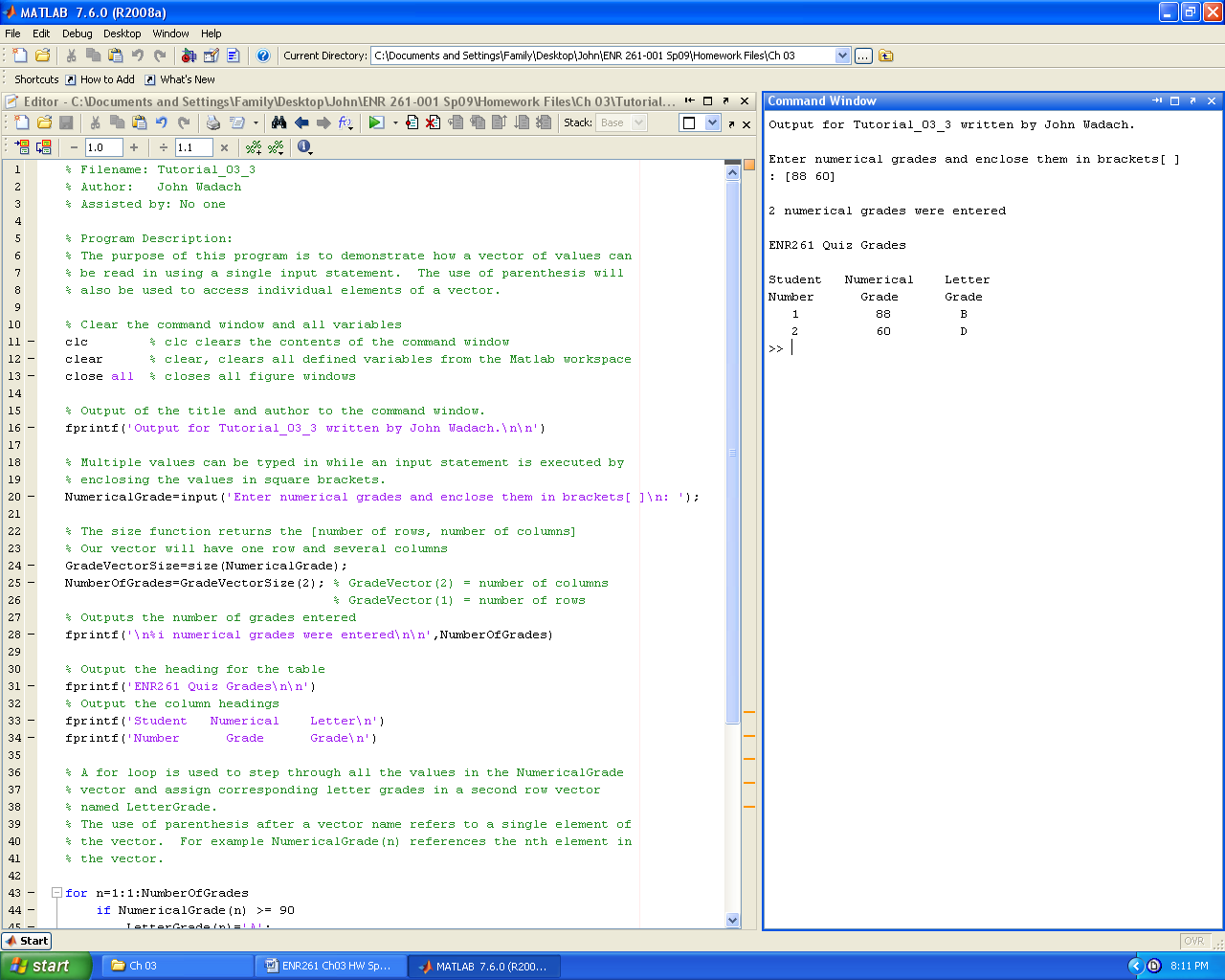
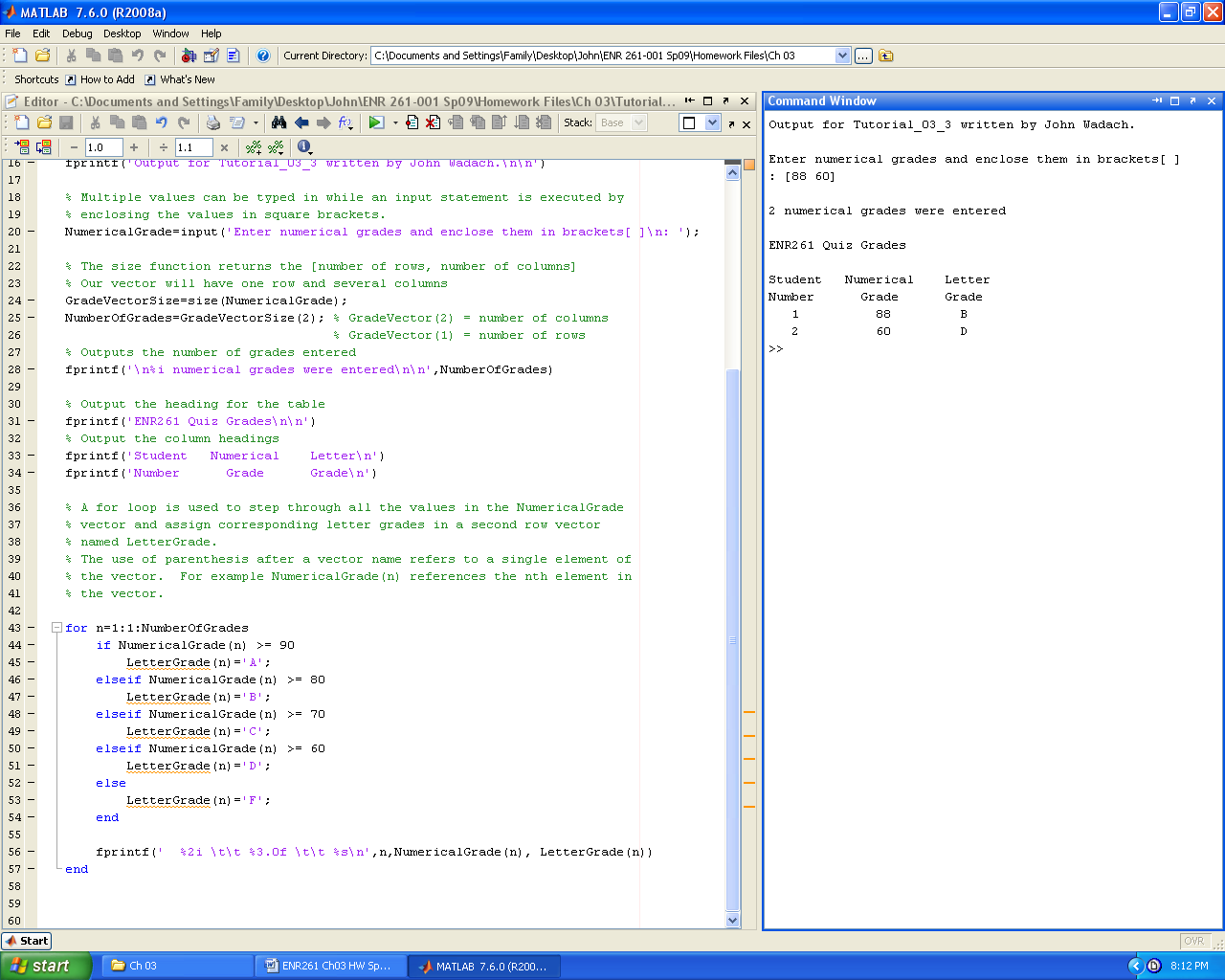
Required File Name: **Tutorial\_03\_2.m**

****

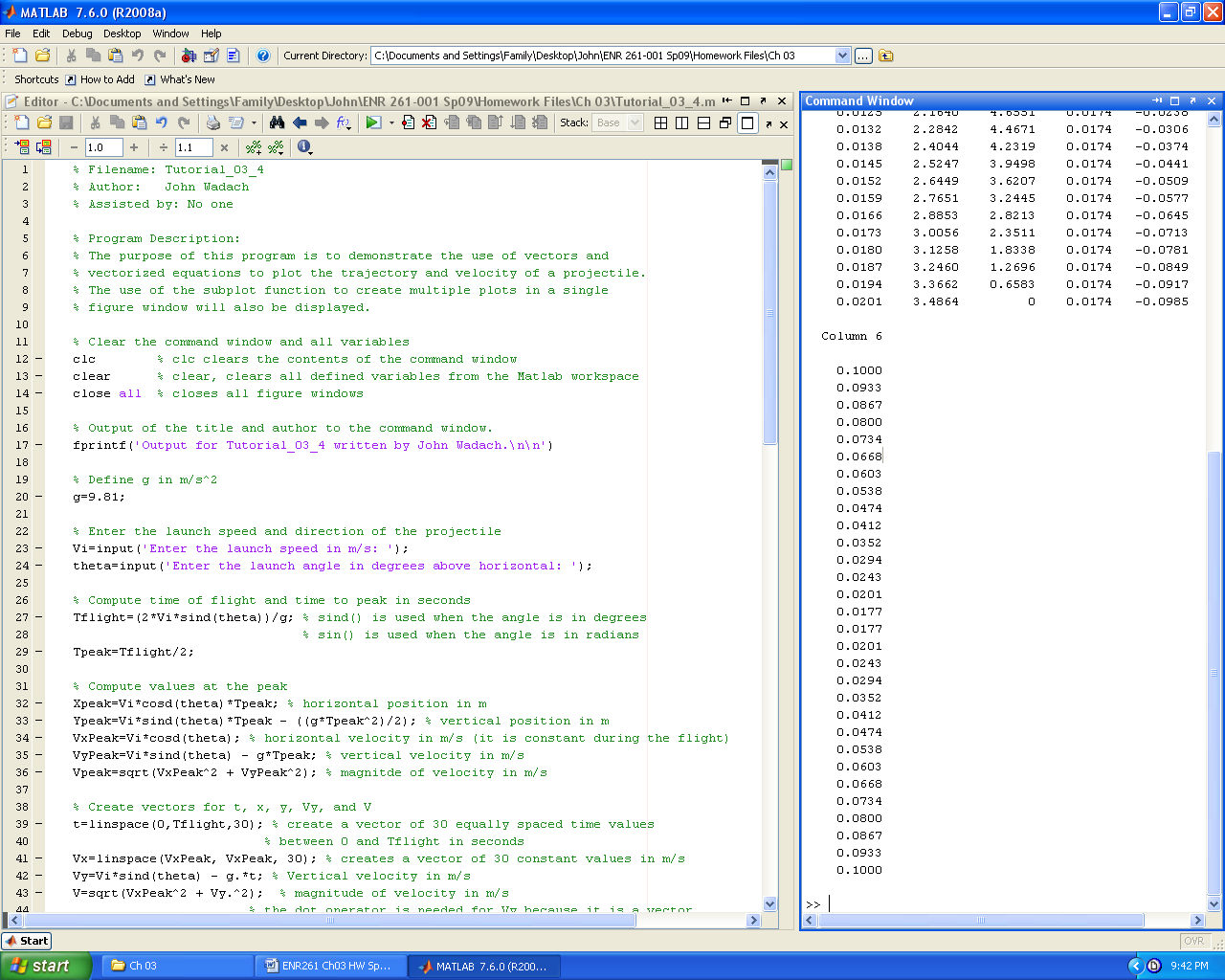
**Continued on the Next Page**

****

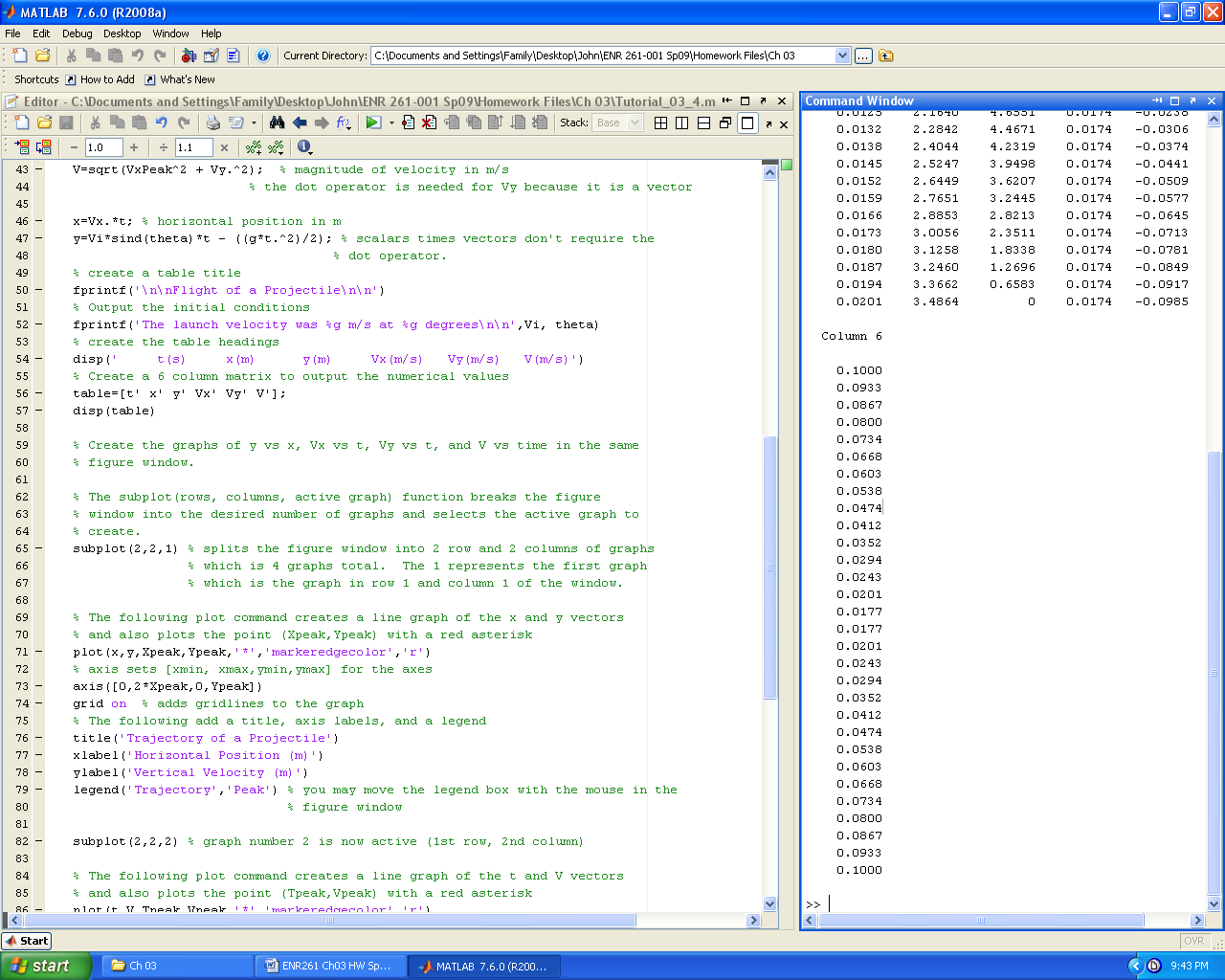
Required File Name: **Tutorial\_03\_3.m**

** **

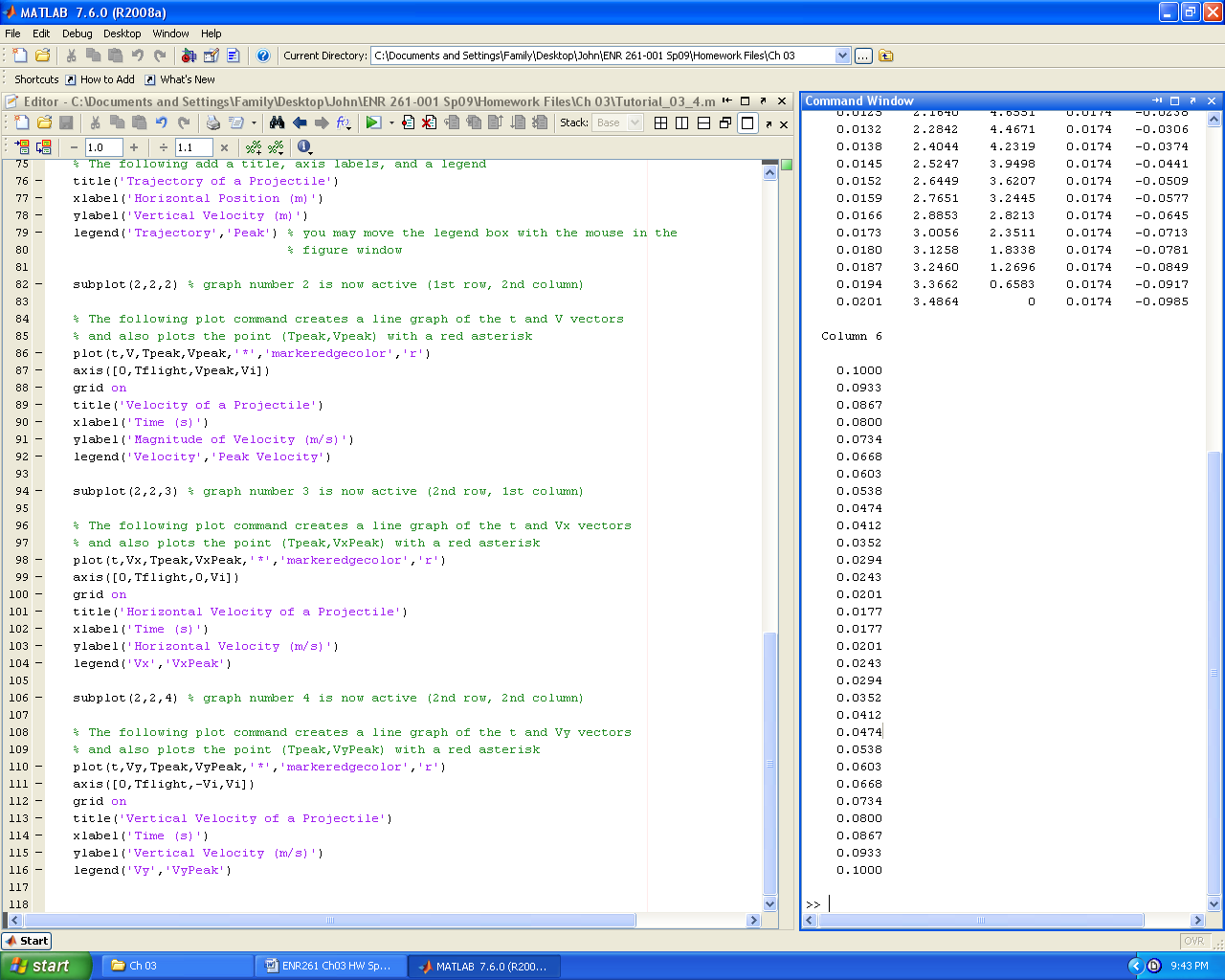
Required File Name: **Tutorial\_03\_4.m**

****

**Continued on the Next Page**

****

**Continued on the Next Page**

****

**Program\_03\_1**

Create a Matlab program to compute the magnitude and phase angle of the impedance (Z) of an RLC series circuit.

Given:

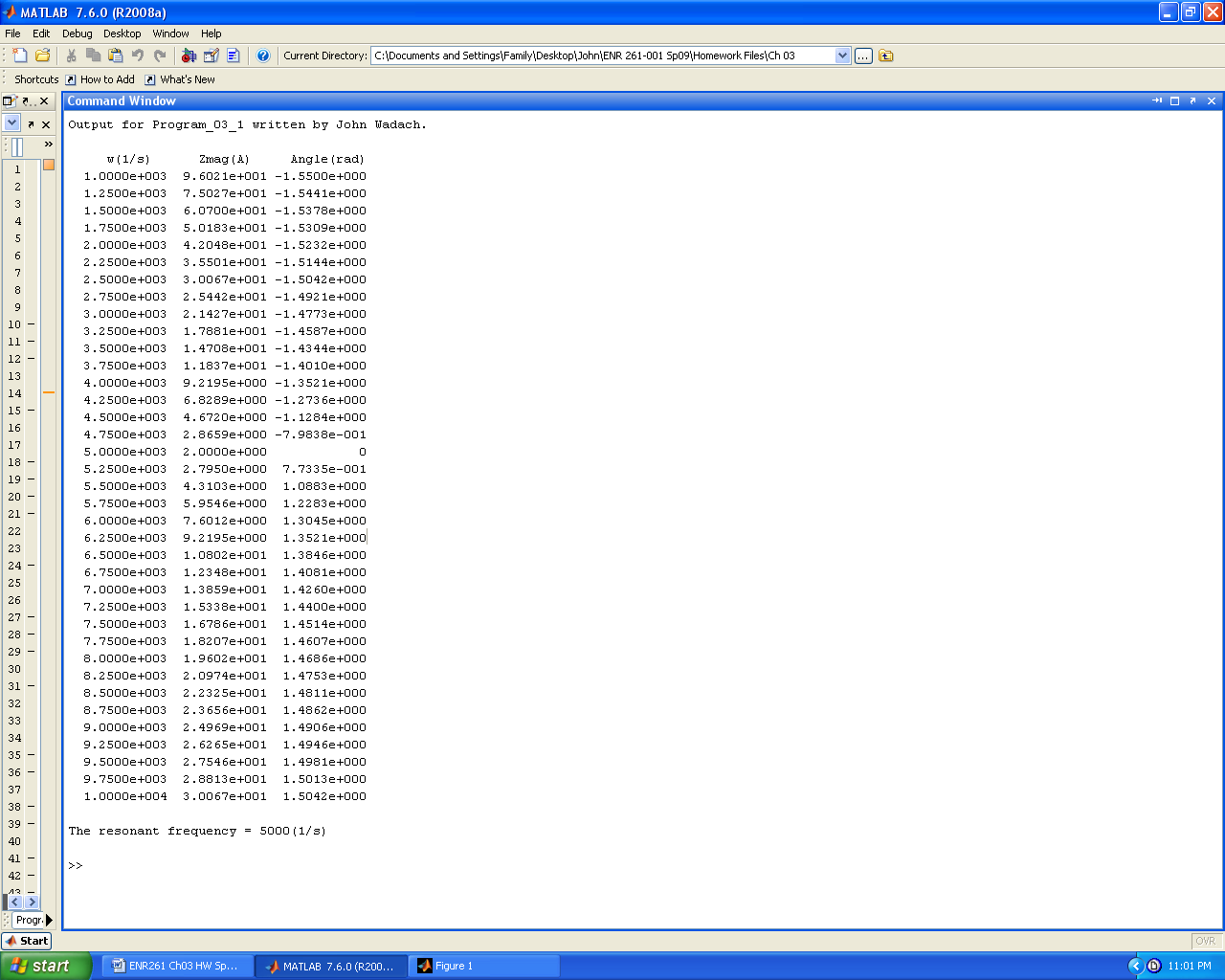
R = 2 (resistance in ohms)

L = 4e-3 (inductance in henries)

C = 10e-6 (capacitance in farads)

Z = R + ((w\*L) - (1/(w\*C)))\*i Impedance in V/A

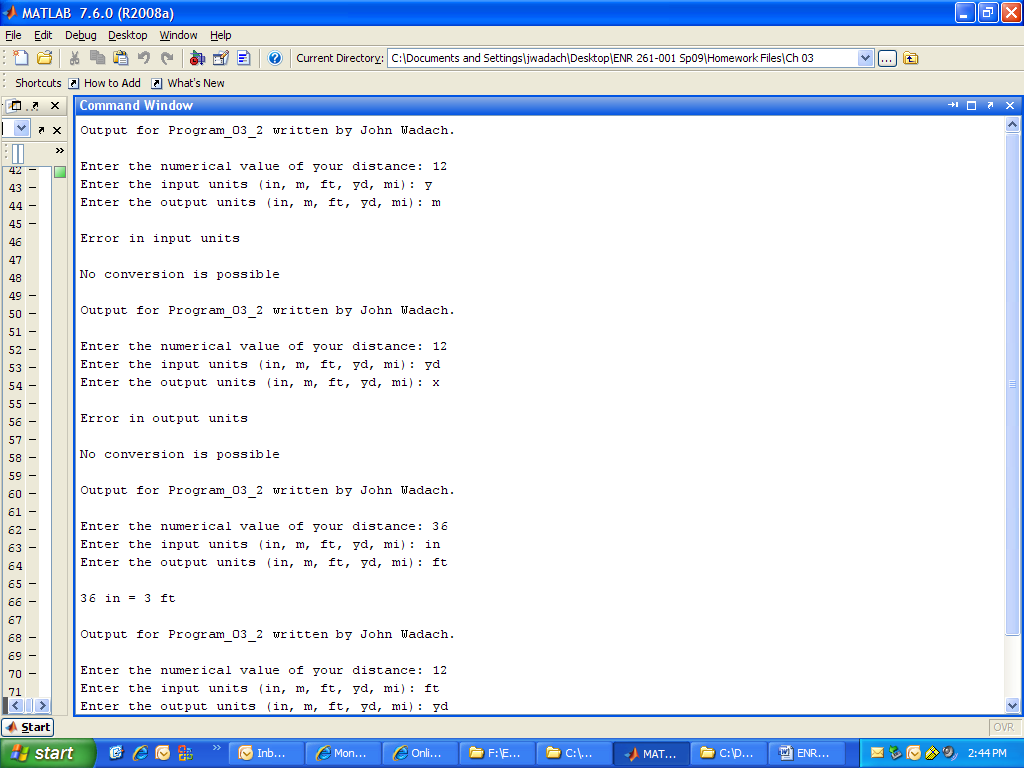
* Create a vector of angular frequencies from 1,000 1/s to 10,000 1/s in steps of 250 1/s.
* Create a Z vector for each frequency
* Create a vector containing the magnitude of Z for each frequency
* Create a vector containing the phase angle of Z for each frequency
* Compute the resonant frequency w0 = 1/sqrt(L\*C)
* Compute the magnitude and phase angle of Z at the resonant frequency
* Present your results in a table and graphs (in the same figure window) as on the next page

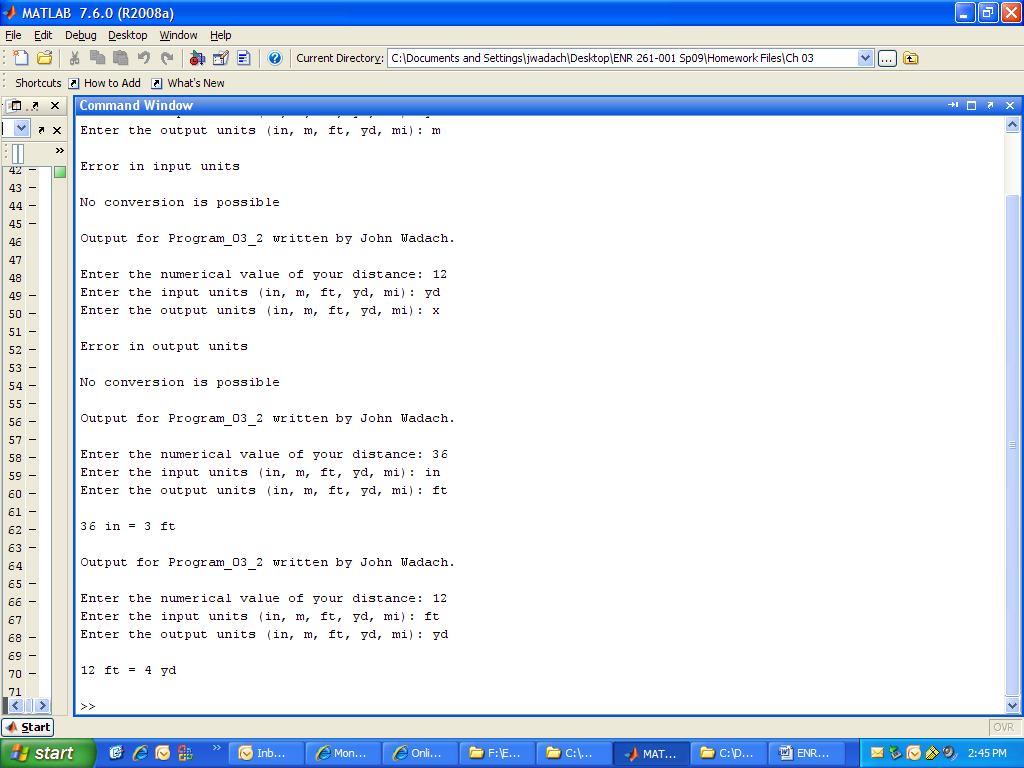
 

**Program\_03\_2**

Create a Matlab program to convert from one unit of distance into another by incorporating the following program elements.

* Look up the conversion factors to create the following variables: InchesPerMeter, InchesPerFoot, InchesPerYard, InchesPerMile.
* Ask the user to input the distance to be converted along with the input units and the output units. The allowable units are in, m, ft, yd, mi.
* When inputting the units include the ‘s’ specifier so that the user does not have to enclose the units in apostrophes.
* Use a switch-case-otherwise construct to convert the inputted number from inputted units into inches. If the input units are not in the allowed list then create an error message and set inches to 0.
* Use a second s switch-case-otherwise construct to convert the distance from inches into the output units. If the input units are not in the allowed list then create an error message.
* If both the input and output units are from the allowable list then output the distance in both the input units and the output units.
* A sample of the output is shown below. When you run your program you may wish to either use the **diary** command or disable the **clc** command so that you can demonstrate thorough testing of your program.





**Program\_03\_3**

Create a Matlab program to track the velocity an object shot straight down (+ velocity) or straight up (- velocity) under the influence of gravity and air drag by incorporating the following program features.

* Define the following constants

g = 9.81; % acceleration of gravity in m/s^2

b = 0.20; % drag coefficient in 1/s

dt = 0.5; % time increment in seconds

tol = 0.1; % in m/s used to stop the iterative process

Vt = g/b; % Terminal Velocity in m/s

n = 1; % index value used to step through the vectors

t(n)=0: % sets the first value of the time vector t0 0 sec

* Have the user input the initial velocity in m/s into a vector Vy(n)
* Compute the initial acceleration using a(n) = g – b\*Vy(n)
* Print the table headings and the row with t(n), v(n) and a(n) with n=1.
* Use a while loop to compute and output the time, velocity, and acceleration of the object. Continue this process until the velocity is between Vt – tol and Vt + tol.
* Inside the loop compute new values for the arrays by incrementing n by one and computing the new values of t(n), Vy(n), and a(n). Recall that the previous value of acceleration a(n-1) , is used in computing the current velocity Vy(n).

Vy(n) = V(n-1) + a(n-1)\*dt

* Create graphs of Vy versus t and a versus t on separate graphs in the same figure window.
* Run your program three times with values of initial velocity of 0, downward velocity greater than Vt, and upward with a velocity greater than Vt.
* A sample of the output is shown on the next page. Only a portion of the table is shown.

