12.010 Homework #5 Solution **Due Thursday, December 01, 2011**

Question (1): *(25-points) (a) Write a Matlab M-file which generates a table of error function (erf) and its derivatives for real arguments (z) between -3 and 3 in steps of 0.25. The error function is defined by the equation below (but is rarely evaluated by performing the integration).*

$$\operatorname{erf}(z) \equiv \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2}\, dt.$$

(see http://mathworld.wolfram.com/Erf.html for information the error function)
The values in the table should be given with 5 decimal places. The table should have headers explaining what the columns are. Explain how you designed the M-file and give an example of the output.
(b) How would you change this M-file if 10 significant digits were required?
Matlab M-file should also be supplied

Solution:

The solution to this problem is easy in Matlab because the Erf function is built in. In the solution we use Matlab's vector operations to form the entries for the table in one single line. By using the correct transpose on the results we are able to print the table with one print statement. We also added to the solution plots of the function and its derivative.
The solution is available at
http://geoweb.mit.edu/~tah/12.010/HW05_01_2011.m

Below is shown in the output from the program for 5 significant digits. To increase the number of significant digits, the format for the table simply needs to be changed. All of the calculations were done with double precision and therefore no change to the variables needs to be made.

```
12.010 HW05 Q01: Table of Erf and its derivative
-------------------------------------
|  Arg x  |    Erf(x)   |  d(Erf)/dx |
-------------------------------------
|  -3.000 |   -0.99998  |    0.00014 |
|  -2.750 |   -0.99990  |    0.00059 |
|  -2.500 |   -0.99959  |    0.00218 |
|  -2.250 |   -0.99854  |    0.00714 |
|  -2.000 |   -0.99532  |    0.02067 |
|  -1.750 |   -0.98667  |    0.05277 |
|  -1.500 |   -0.96611  |    0.11893 |
|  -1.250 |   -0.92290  |    0.23652 |
|  -1.000 |   -0.84270  |    0.41511 |
|  -0.750 |   -0.71116  |    0.64293 |
|  -0.500 |   -0.52050  |    0.87878 |
```

-0.250	-0.27633	1.06001
0.000	0.00000	1.12838
0.250	0.27633	1.06001
0.500	0.52050	0.87878
0.750	0.71116	0.64293
1.000	0.84270	0.41511
1.250	0.92290	0.23652
1.500	0.96611	0.11893
1.750	0.98667	0.05277
2.000	0.99532	0.02067
2.250	0.99854	0.00714
2.500	0.99959	0.00218
2.750	0.99990	0.00059
3.000	0.99998	0.00014

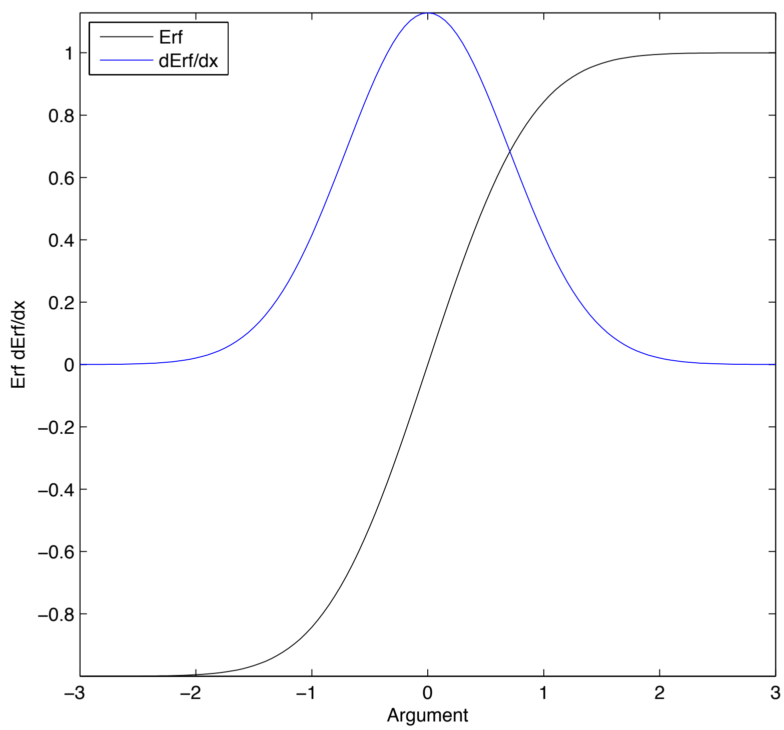

Figure 1. Plot of the Erf function and its derivative.

Question (2): *(25-points).*

Write an M-file that reads your name in the form <first name> <middle name> <last name> and outputs the last name first and adds a comma after the name, the first name, and initial of your middle name with a period after the middle initial. If the names start with lower case letters, then these should be capitalized. The M-file should not be specific to the lengths of your name (ie., the M-file should work with anyone's name.
As an example. An input of

thomas abram herring
would generate:
Herring, Thomas A.

Solution:

This solution is also easy in Matlab using the tolower and toupper functions. The code also detects whether two or three names have been given and changes the output accordingly.

Solution is available at
http://geoweb.mit.edu/~tah/12.010/HW05_02_2011.m

```
----------------------------------------
12.010 HW 05 Q2:
Enter Name (first middle last) thomas abram herring
Herring, Thomas A.
```

Question (3): *(50-points) Write a Matlab M-file that will compute the motion of a bicyclist and the energy used cycling along an oscillating, sloped straight-line path. The path followed will be expressed as*

$$H(x) = Sx + A\sin(2\pi x/\lambda) + B\cos(2\pi x/\lambda)$$

where H(x) is the height of the path above the starting height, S is a slope in m/m, A and B are amplitudes of sinusoidal oscillations in the path. The wavelength of the oscillations is λ*. The forces acting on the bicycle are:*

$$\text{Wind Drag} \quad F_d = 1/2 A_r C_d \rho V^2$$
$$\text{Rolling Drag} \quad F_r = M_r g C_r$$

where A_r *is the cross-sectional area of the rider,* C_d *is the drag coefficient, r is the density of air and V is the velocity of the bike. For the rolling drag,* M_r *is the mass of the rider and bike, g is gravitation acceleration and* C_r *is rolling drag coefficient.*

The bicyclist puts power into the bike by pedaling. The force generated by this power is given by

$$\text{Rider force} \quad F_r = P_r / V$$

where F_r *is the force produced by the rider,* P_r *is power used by the rider and V is velocity that the bike is traveling (the force is assumed to act along the velocity vector of the bike). Your M-file can assume that the power can be used at different rates along the path. The energy used will be the integrated power supplied by the rider. Assume that there is maximum value to the rider force.*

Your code should allow for input of the constants above (path and force coefficients). The M-file can assume a constant power scenario and constant force at low velocities.

As a test of your M-file use the following constants to compute:

(a) *Time to travel and energy used to travel 10 km along a path specified by S=0.001, A=5.0 m, B=0.0 m and λ= 2km, with constant power use of P_r =100Watts and a maximum force available of 20N.*
(b) *The position and velocity of the bike tabulated at a 100-second interval.*
(c) *Add graphics to your M-file which plots the velocity of the bike as a function of time and position along the path.*

Assume the following values

$C_d = 0.9$

$C_r = 0.007$

$A_r = 0.67\ m^2$

$\rho = 1.226\ kg/m^3$

$g = 9.8\ m/s^2$

$M_r = 80\ kg$

In this case, the Matlab M-file will not be of the type used for fortran and C/C++. Look at the documentation on ODExx where xx is a pair of number for Ordinary Differential Equation solutions.

Your answer to this question should include:

(a) *The algorithms used and the design of your M-file*
(b) *The Matlab M-file with your code and solution (I run your M-file).*
(c) *The results from the test case above.*

Solution:

The solution to this problem is also quite easy in Matlab. We use the OED 45 1st order differential equation solver. To use this function we also have a function that computes the acceleration at the bike and another function that is the event detector that determines when the bike reaches 10 km distance. For ease of coding, the variables that control the characteristics of the bike are saved as globals. We also use an input dialog box to both set the defaults and allow the user to change the values at runtime.
Also included in the solution is an animation and plot of the motion of the bike. To have the animation runs smoothly we change the output interval of the differential equation solver so that we generate more points then those that given in the standard 100 seconds separated values. There are some interesting bugs in Matlab for plotting the velocity vectors. The quiver option is used for this plot but in the current release of Matlab the arrowheads are extremely large and are turned off in the plots. Also the automatic scaling of the vectors does not generate reasonable length vectors, and so in these plots we manually set the scale factor.

To compute the energy used during a bike ride, we simply added the derivative of the energy to the differential equations. This way the total energy was integrated automatically in the solution and could be output to the table as shown below.

The solution M-files are available at:
http://geoweb.mit.edu/~tah/12.010/HW05_03_2011.m
http://geoweb.mit.edu/~tah/12.010/bikeacc.m
http://geoweb.mit.edu/~tah/12.010/bikehit.m
http://geoweb.mit.edu/~tah/12.010/banimate.m

The table and figures below show the results for the default solution.

```
----------------------------------------------------------------------
12.010 HW 05 Q 3: Bike simulation
----------------------------------------------------------------------
```

Time (sec)	X pos (m)	Z pos (m)	X Vel (m/s)	Z Vel (m/s)	Energy (Joules)
0.000	0.0000	0.0000	0.0000	0.0000	0.00
100.000	79.9607	1.3228	1.4894	0.0242	1599.43
200.000	306.7804	4.4136	3.3662	0.0335	6136.26
300.000	826.7200	3.4173	6.7979	-0.0845	15406.88
400.000	1513.7483	-3.4812	5.8778	0.0099	25406.88
500.000	1951.5589	1.1930	2.9595	0.0489	33908.48
600.000	2202.3092	5.1692	2.6551	0.0362	38924.12
700.000	2608.2191	7.3210	5.8263	-0.0247	46828.92
800.000	3292.5203	-0.6831	6.8901	-0.0588	56828.92
900.000	3834.4703	1.3482	3.8829	0.0568	66438.93
1000.000	4117.1190	5.9128	2.3536	0.0368	72092.64
1100.000	4427.6762	9.2952	4.5307	0.0206	78304.19
1200.000	5045.7397	4.3271	7.2502	-0.1055	88249.44
1300.000	5681.1563	1.4650	4.9724	0.0471	98248.16
1400.000	6034.8791	6.5772	2.4814	0.0412	105321.74
1500.000	6293.6398	10.2743	3.3172	0.0348	110497.49
1600.000	6802.7335	9.7025	6.7096	-0.0791	119675.95
1700.000	7492.6695	2.4890	5.9908	0.0038	129675.95
1800.000	7941.2097	7.0155	3.0363	0.0499	138331.57
1900.000	8193.2972	11.0363	2.6051	0.0362	143373.96
2000.000	8588.5246	13.3858	5.7243	-0.0190	151121.55
2100.000	9268.9477	5.5189	6.9633	-0.0657	161121.55
2200.000	9821.1684	7.1456	3.9935	0.0571	170809.56
2255.202	10000.0000	9.9890	2.6506	0.0443	174386.65

```
----------------------------------------------------------------------
Time : 2255.202, Velocity 2.6506 0.0443
Energy used:    174386.65 Joules,    41.64 kcals

To show animation: banimate(ta,ya)
>> banimate(ta,ya)
```

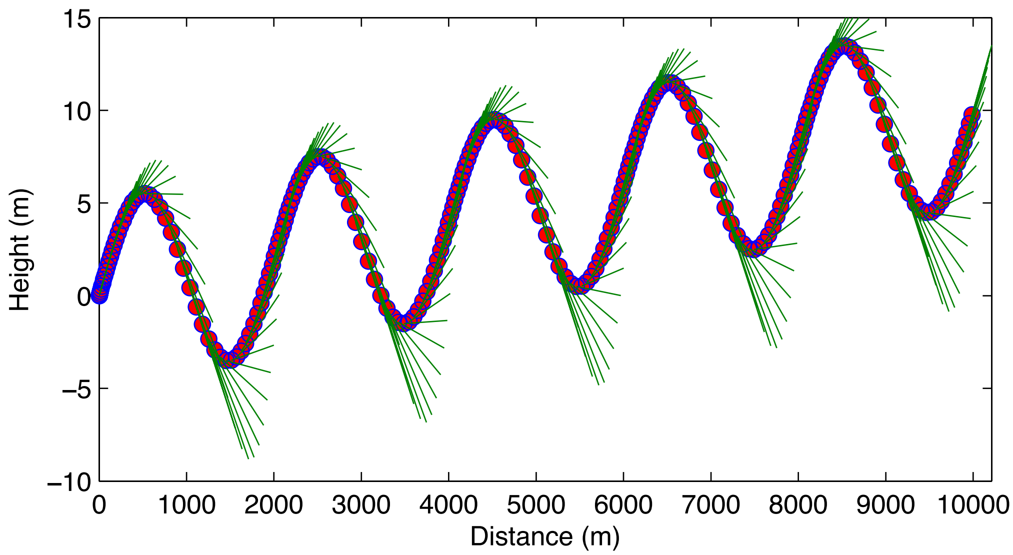

Figure 2: Trajectory of the bike along with vectors showing its velocity (green lines). Due to poor scaling in Matlab, no arrowheads are shown.

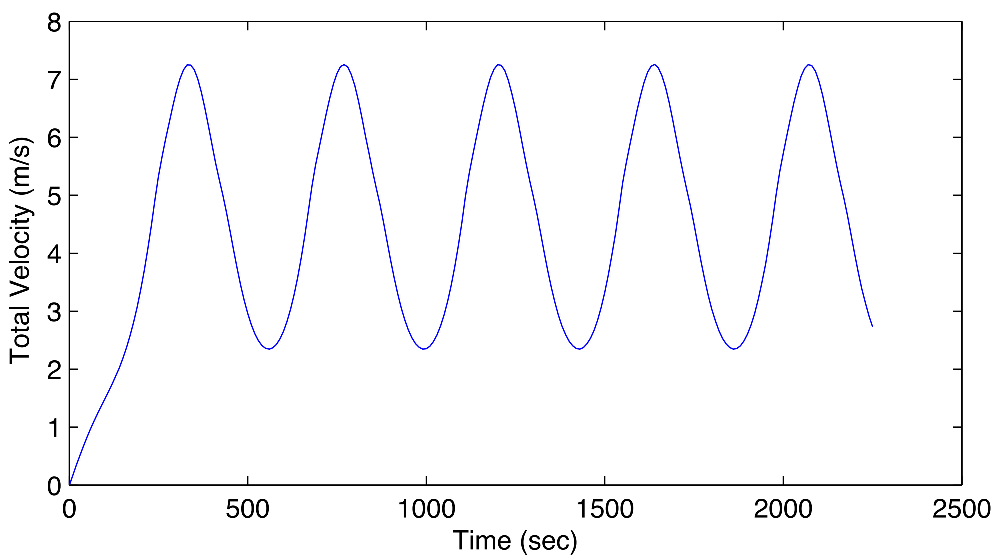

Figure 3: Velocity as a function of time for the bike.

12.010 Computational Methods of Scientific Programming
Fall 2011