MAE 8 - Winter 2015
Homework 6

Instructions: Follow the homework solution template. Put all answers in a MATLAB script named **hw6.m**. For this homework, you will need to submit multiple files. Create a zip archive named **hw6.zip**. The zip archive should include the following files: **hw6.m**, **bilinear.m**, **projectile1D.m**, **figure1.png**, **temperature.mat** and **Astring.mat**. Submit **hw6.zip** through TED before 9 PM on 02/19/2015. Use double precision unless otherwise stated.

Problem 1:

In engineering there are many occasions in which the following scenario occurs: A measured quantity (say a two-dimensional temperature field) is known at specific x-y locations yet a value is needed at a point P = (x, y) that is between points at which the temperature was measured. One way to estimate the temperature of point P from the surrounding locations is bilinear interpolation. Details of how bilinear interpolation is implemented are given at `http://en.wikipedia.org/wiki/Bilinear_interpolation`.

In this exercise, you will make use of local (sub) functions. The primary function and local functions all should be stored in MATLAB file **bilinear.m**. All functions should detail on how to call and the inputs and outputs. Do not write three separate files for this exercise.

The primary function should have the following header: **function [Tp] = bilinear(x, y, T, Px, Py)**. The inputs are one-dimensional x-grid **x** and y-grid **y**, the two-dimensional temperature **T**, and the location **Px** and **Py** of target point P. The output is the single value of temperature **Tp** at the point P. This function calls the following two local (sub) functions to complete the task.

The first local (sub) function **bound_target** should find the indices **iloc** in vector **x** and **jloc** in vector **y** where x(iloc) is closest and smaller than Px and y(jloc) is closest and smaller than Py. This local function should have the following header: **function [iloc, jloc] = bound_target(x, y, Px, Py)**. If the target point is not bounded by the grid, the function should output an error message and return (use function **return**).

The second local function should have the following header: **function [Tp] = interp_target(x, y, iloc, jloc, T, Px, Py)** The function takes input grid vectors **x** and **y**, indices **iloc** and **jloc**, the temperature **T**, the location **Px** and **Py**, and returns **Tp** as the output. This is where the bilinear interpolation takes place.

Download the file **temperature.mat** from TED and load it to MATLAB. The file has one dimensional x-grid **x** and y-grid **y** , and the two-dimensional temperature field **T**.

(a) Set **p1a=evalc('help bilinear')**.
(b) Set **p1b=evalc('help bilinear>bound_target')**.
(c) Set **p1c=evalc('help bilinear>interp_target')**.
(d) Let Px = -7.1 and Py = 7.5, and get the interpolated temperature and set it to **p1d**.
(e) Let Px = 1.5 and Py = -14.1, and get the interpolated temperature and set it to **p1e**.
(f) Let Px = -7.1 and Py = -14.1, and get the interpolated temperature and set it to **p1f**.

(g) Let Px = 3.2 and Py = -1.2, and get the interpolated temperature and set it to **p1g**.

(h) Let Px = 3.1 and Py = -1.1, and get the interpolated temperature and set it to **p1h**.

(i) Create **xfine** and **yfine** grids that have twice the resolution of the initial x and y grids. Then use two **for** loops to interpolate the temperature onto all the new grid points and store the new two-dimensional temperature field as **Tfine**. Use command **surf(Tfine)** to plot the fine temperature field in a figure. Make sure that the surface plot shows when your script is run. Set **p1i='See figure 1'** and save the figure as **figure1.png**.

Problem 2:

A ball at a height Zo is thrown upward at an initial vertical velocity Wo. In this exercise, you are to explore motion of the ball numerically using Euler's method. The motion is described by the following differential equations:

$$\begin{aligned} \frac{dZ}{dt} &= W, \\ \frac{dW}{dt} &= -g, \end{aligned}$$

where **t** is time, **Z** is height, **W** is vertical velocity of the ball and **g** is gravity. Using Euler's method, the equations can be approximated by

$$\begin{aligned} Z^{n+1} &= Z^n + W\Delta t, \\ W^{n+1} &= W^n - g\Delta t, \end{aligned}$$

where superscript n denotes variables at current time, superscript n+1 denotes variables at time that is Δt ahead.

Write function **projectile1D.m** to numerically solve for the motion of the ball. The function should have the following header: **function [T, Z, W] = projectile1D(Zo,Wo, Tf, dt)**. The inputs are the initial height of the ball **Zo**, the initial vertical velocity Wo, the duration of the motion **Tf** and the time step **dt**. The outputs are vectors **T**, **Z** and **W** which are the time, the height and the vertical velocity of the ball, respectively. Give the function a description. Set gravity to be $9.81 m/s^2$ in the following exercises.

(a) Set **p2a=evalc('help projectile1D')**.

(b) Let **Zo = 500 m** and **Wo = 0 m/s**. Create a vector **time = [0:0.01:10]**. Compute the analytical solution of this motion Z(t) for a duration of 10 s and put the answer in **p2b**.

(c,d,e) Use function **projectile1D** to get the time, the height, and the vertical velocity of the motions for 10 s. Set the answers to **p2c**, **p2d**, and **p2e**, respectively. Use 1-second time step.

(f,g,h) Repeat the step above with 0.01-second time step. Put the time, the height, and the vertical velocity into **p2f**, **p2g**, and **p2h**, respectively.

(i) Make a figure to plot height versus time. The figure should include 3 plots: one with the analytical solution in part (b), one with the 1-second time step in parts (c-d) and one with the 0.01-second time step in parts (f-g). Set **p2i='See figure 2'**.

(j) As the time step is reduced, does the height Z get closer to the analytical solution ? Give the answer in the form **p2j='...'**.

(k) Now keep **dt = 0.01 s**, **Zo = 500 m** and **Tf = 10 s**. Compute the maximum height that the ball can reach in different cases where the initial vertical velocity **Wo** increases linearly from 0 m/s to 50 m/s with an increment of 5 m/s. Put the answer in **p2k**.

(l) How does the maximum height in part (k) vary with the initial vertical velocity **Wo** ? Give the answer in the form **p2l='...'**.

Problem 3: Download the file **Astring.mat** from TED and perform the following exercise. This file contains a string named **Astring**.

(a) Search for any instances of 'matlab'and replace them to 'MATLAB'. Put the answer in **p3a**.

(b) How many times does 'matlab 'occur in the string ? Put the answer in **p3b**.

(c) Determine the percentage of characters that correspond to letter in the string. Your script should report the answer to **p3c** as a decimal (i.e. 99% is 0.99) and in double precision, do not use a string.

(d) Remove the characters prior to the first letter 'm'in the string and put the answer in **p3d**.

(e) Replace all letters in the string with spaces and put the answer in **p3e**.