Math-3315 & CSE-3365, programming exercises

August 21, 2012

Exercise 1. Study and experiment with the two programs

http://faculty.smu.edu/yzhou/Teach/2012F/lapla.m

http://faculty.smu.edu/yzhou/Teach/2012F/laptest.m,

try to understand the matlab commands and syntax involved.

Exercise 2. For `laptest.m` listed above, try calling it as

```
[ cpuvec ] = laptest(500, 250, 3500);
```

(If your computer is relatively fast and has relatively large memory, you may try larger inputs such as `cpuvec=laptest(1000, 1000, 9000);`).

Use the info stored in the output vector `cpuvec` to make a plot of how many times the 2-loop method is slower than the 1-loop method.

(hint: plot the vector `cpuvec(:,1) ./ cpuvec(:,2)`)

Exercise 3. (This exercise needs Matlab, not octave) Experiment with the program

http://faculty.smu.edu/yzhou/Teach/2012F/svdimage.m .

That is, calling it as

```
A = load('clown.mat');    svdimage(A, 1e-2)
```

or

```
A = imread('http://www.mathworks.com/moler/ncm/fern.png');
svdimage(A, 1e-3, 5);
```

Then try the same program on a photo image of your own (you may use your SMU student ID photo) and see how your image evolve under the SVD compression.

```
%need to replace 'myphoto.jpeg' by your image filename
A = imread('myphoto.jpeg');
svdimage(A, 1e-3, 2);
```

Exercise 4. Matlab/Octave has a built-in function `magic.m`. The command `magic(n)` computes a size $n \times n$ magic square, which contains the integers 1 to n^2 exactly once, with each row sum, column sum, diagonal sum, and anti-diagonal sum all equal to each other.

Magic squares were known in China over 2000 B.C. The 3×3 magic square dates back to about 2850 B.C. and was known as *Lo Shu*. The 4×4 magic square is sometimes called the Dürer's square because of his famous engraving Melencolia I (Figure 1).

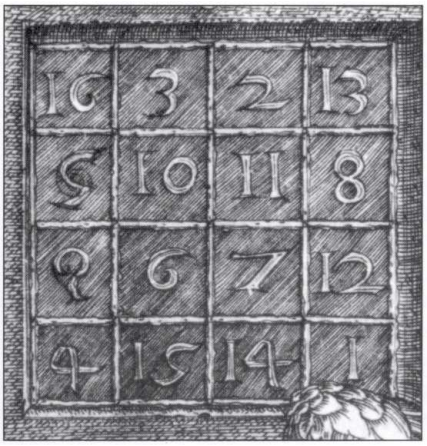

Figure 1: Albrecht Dürer's engraving Melencolia I.

Study the `magic.m` (Figure 2) code and appreciate the wisdom inside this succinct script. (It solved a pretty hard problem from ancient times.) Notice the documentation, comments, conditional tests, and other useful matrix manipulations.

Write a Matlab function that achieves the following task:

```
function  [ out ]= magic_square_test(M);
%
% This code tests if an input square matrix is a magic-square.
% The output variable is of boolean type:
%     if M is magic-square, then out='true', else, out='false'.
%
```

Please use loop to go through the rows and columns of M to get the row sums and column sums. For the diagonal sum, use `sum(diag(M))`; for the anti-diagonal sum, use `sum(diag(flipud(M)))`. Compare the sums, exit and return with out='false' when the you encounter the first two sums that are not equal.

For example, if you call your code as

```
[ out ]= magic_square_test(magic(10))
```

the output variable should be the correct out='true'.

```
function M = magic(n)
%MAGIC  Magic square.
%   MAGIC(N) is an N-by-N matrix constructed from the integers
%   1 through N^2 with equal row, column, and diagonal sums.
%   Produces valid magic squares for all N > 0 except N = 2.

%   Copyright 1984-2010 The MathWorks, Inc.
%   $Revision: 1.7 $  $Date: 2012/08/23 03:40:05 $

n = floor(real(double(n(1))));

% Odd order.
if mod(n,2) == 1
   [J,I] = meshgrid(1:n);
   A = mod(I+J-(n+3)/2,n);
   B = mod(I+2*J-2,n);
   M = n*A + B + 1;

% Doubly even order.
elseif mod(n,4) == 0
   [J,I] = meshgrid(1:n);
   K = fix(mod(I,4)/2) == fix(mod(J,4)/2);
   M = reshape(1:n*n,n,n)';
   M(K) = n*n+1 - M(K);

% Singly even order.
else
   p = n/2;
   M = magic(p);  %recursive call to magic.m
   M = [M M+2*p^2; M+3*p^2 M+p^2];
   if n == 2, return, end
   i = (1:p)';
   k = (n-2)/4;
   j = [1:k (n-k+2):n];
   M([i; i+p],j) = M([i+p; i],j);
   i = k+1;
   j = [1 i];
   M([i; i+p],j) = M([i+p; i],j);
end
```

Figure 2: MathWorks distributed `magic.m` code.

Comment: There are some more succinct way to test if the sums of rows, columns, and diagonals are equal or not, without using loop. For example

```
function  [ out ]= magic_square_test(M);
%
% first, test if the sums of rows, columns, and diagonals  are equal or not
%
error = norm(diff([ sum(M), sum(M'), sum(diag(M)), sum(diag(flipud(M)))]));
if error ~= 0,
   out = 'false';
else
   % second, test if M satisfies other conditions of a magic square,
   % such as if it contains integers from 1 to n^2 only once.
   % (for simplicity, you may assume that these conditions are satisfied)
   out = 'true';
end
```

Exercise 5. Download the following Matlab m file

http://faculty.smu.edu/yzhou/Teach/2012F/scwave.m

and experiment with it. This script simulates the wave structure installed before the Meadow's museum. (You don't need to understand the method used, only need to notice the Matlab syntax and experiment with different inputs to the matlab function.)

A more challenging exercise:
Can you write a different script that simulates the wave structure? Hint, the wave structure may be essentially represented by the following 3-d surface

$$z(x, y) = \beta * y * \sin(\alpha * x),$$

where α, β are scaling factors (scalars). You may set $\alpha = \beta = 1$ for simplicity.

Exercise 6. Copy and paste the following code into a script file, then run it in Matlab.

```
Z = peaks(50);
surf(Z);
axis tight
set(gca,'nextplot','replacechildren');
m = 30;
for j =  1 : m
    surf(cos(2*pi*j/m)*Z, Z)
    Frame(j) = getframe;
end
movie(Frame, 5)
```

Use `doc` to understand what each command performs, then add comments to your script to make this script understandable (hopefully even by someone who does not know Matlab). This practice trains your ability in understanding the Matlab documents and in making your code understandable by others.