M647, Spring 2012, Assignment 1, due Friday Jan. 27

1. [10 pts] For these plotting assignments, use *plot* rather than *ezplot* or *fplot*. You should write a single MATLAB script M-file that produces all your plots, and you should turn this in along with a separate plot for each part. If you put the *figure* command above each plot command MATLAB will create your five plots in five different windows, and you will have access to them all at once.

1a. Plot the function $f(x) = x + \sin x$ for $x \in [0, 2\pi]$. Label your x and y axes and add a title to your plot.

1b. Plot the functions $x(t) = \tan t$ and $y(t) = \cos t$ on a single figure for $t \in (-\frac{\pi}{4}, \frac{\pi}{4})$. Label your x and y axes (i.e., your horizontal and vertical axes, since your horizontal variable will be t) and add a title to your plot. Determine (roughly) from your figure the point of intersection.

1c. Plot the path $(x(t), y(t))$ for $x(t)$ and $y(t)$ as defined in (b), again for $t \in (-\frac{\pi}{4}, \frac{\pi}{4})$. Label your x and y axes and add a title to your plot.

1d. Use MATLAB's built-in help to find information about the *legend* function, and use this function to add a legend to your plot from (b).

1e. MATLAB's built-in functions *text* and *gtext* can both be used to add text to a plot at a particular location. Use the command *text(0,-.2,'tan(t)')* to label the curve $x(t) = \tan t$ on your plot from (b). (This command places the text *tan(t)* so that its first character is at the designated position $(0, -.2)$). Likewise, use the command *gtext('cos(t)')* to place the text *cos(t)* in a reasonable location relative to the curve $y(t) = \cos t$. (This command will open up your figure window and allow you to select the location of the text with a mouse click.)

2. [10 pts] An amortized loan is one in which an initial principal P_0 is borrowed at some interest rate r, and payments with value v are made at equal time intervals, often monthly. (For example, car and house payments typically work this way.) Here, we are assuming r denotes interest rate for the payment period, so for example if the annual interest rate is .05 and the payments are monthly, then $r = .05/12$. (To be clear, it's generally assumed in these situations that when we say annual rate r_A we mean monthly rate $\frac{r_A}{12}$, even though monthly growth at rate $\frac{r_A}{12}$ isn't quite the same as annual growth at rate r_A.) If we let P_t denote the remaining balance at time t, then we clearly have the difference equation

$$P_{t+1} = P_t(1 + r) - v,$$

which can be solved exactly as

$$P_t = (1 + r)^t P_0 - v\frac{(1 + r)^t - 1}{r}.$$

2a. Write a function M-file that takes as input the values r and P_0, and the duration of the loan, and returns the payment value v required to pay off the loan.

2b. If a car loan is $P_0 = 20{,}000$ for five years at annual rate .05, what must the monthly payment v be?

3. [10 pts] A bilinear form is a mathematical object over a vector space V that takes two vectors as input and returns a scalar as output. Also, if one of the arguments is fixed, it must be linear as a function of the other. The example we will work with for this problem is the bilinear form

$$B_A(\vec{v},\vec{w}) = \vec{v}^{tr} A \vec{w},$$

where A is any $n \times n$ matrix and $\vec{v}$ and $\vec{w}$ are both $n \times 1$ column vectors (so $\vec{v}^{tr}$ is a $1 \times n$ row vector).

3a. Write a MATLAB function M-file that takes as input an $n \times n$ matrix A, two $n \times 1$ vectors $\vec{v}$ and $\vec{w}$, and returns the value $B(\vec{v},\vec{w})$.

3b. Use your result from (a) to compute $B_A(\vec{v},\vec{w})$ when

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix}; \quad \vec{v} = \begin{pmatrix} -1 \\ -1 \\ 1 \\ 0 \end{pmatrix}; \quad \vec{w} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}.$$

3c. If $B_A(\vec{v},\vec{v}) > 0$ for all $\vec{v}$ not identically 0 we say the matrix A is positive definite. Use your M-file to show that the matrix A from (b) is not positive definite.

3d. An $n \times n$ matrix A is positive definite if and only if its eigenvalues are all positive. Use MATLAB's *eigs* command to verify that A has some non-positive eigenvalues.

4. [10 pts] In this problem we will consider MATLAB objects known as *anonymous* functions, and also a number of additional plot commands. You should write a single MATLAB script M-file that produces all your plots, and you should turn this in along with a separate plot for each part.

4a. Define the function

$$f(x) = x^2 e^{-x^2}$$

by using the command *f=@(x)x^2*exp(-x^2)*. Technically, f is now a *function handle*, which is an easy object to work with. For example, you can now evaluate f in the MATLAB Command Window by typing $f(0)$, $f(1)$ etc. Plot this function on the interval $[0, 2]$ by using the command *fplot(f,[0,2])*.

4b. For implicit relations such as $x^2 + y^2 = 1$, plotting with either *plot* or *fplot* can be clunky, and it's often more convenient to use *ezplot*. As an example, we'll plot the *Folium of Descartes*

$$x^3 + y^3 = 6xy.$$

First, we re-write the equation as $x^3 + y^3 - 6xy = 0$ and define the left-hand side as an anonymous function *f=@(x,y)x.^3+y.^3-6*x.*y*. Now, plot the Folium for $x \in [-1, 4]$ with the command *ezplot(f,[-1,4])*. (Notice that MATLAB automatically sets $f = 0$.)

4c. Use MATLAB's *plot3* command to plot the helix described parametrically by

$$\begin{aligned} x(t) &= \cos t \\ y(t) &= \sin t \\ z(t) &= t, \end{aligned}$$

for $t \in [0, 12\pi]$.

4d. Use MATLAB's *mesh* command to plot the function

$$f(x,y) = x^2 + y^2$$

on the square $[-2,2] \times [-2,2]$. (**Hint.** You'll find MATLAB's help on the *meshgrid* command useful for this problem.)

5. [10 pts] The purpose of this problem is to introduce students to two useful objects in MATLAB: *structures* and *cell arrays.*[1] These objects are designed for roughly the same purpose, to allow storage of different data types in the same variable. As an example, suppose we would like to create a variable *mypoly* that contains as its first element the string text "My favorite polynomial," as its second element the (anonymous function) polynomial $f(x) = x^2 - x - 1$, and as its third element the roots of this polynomial—the larger of which is often referred to as the *golden mean.*

5a. In order to create *mypoly* as a structure, we need to know that the basic format of components of structure variables is *mypoly.component1, mypoly.component2*, etc. Consider the following MATLAB session:

```
>>mypoly.name = 'My favorite polynomial';
>>mypoly.function = @(x) x^2-x-1;
>>mypoly.roots = solve('x^2-x-1');
>>mypoly
mypoly =
name: 'My favorite polynomial'
function: @(x)x^2-x-1
roots: [2x1 sym]
>>mypoly.name
ans =
My favorite polynomial
>>mypoly.function(1)
ans =
-1
>>mypoly.roots(1)
ans =
1/2 - 5^(1/2)/2
>>mypoly.roots(2)
ans =
5^(1/2)/2 + 1/2
```

Notice in particular that the component *mypoly.function* takes input like an anonymous function (which is what it is), while the component *mypoly.roots* interprets input as vector

[1]For example, we will encounter *structures* when we carry out linear regression in MATLAB and when we solve boundary value problems, and *cell arrays* are required for function M-files that take a variable number of inputs.

components. Create a MATLAB structure that contains as its first component the string text "My favorite matrix," as its second component the matrix

$$M = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 4 & 9 & 16 \\ 1 & 8 & 27 & 64 \end{pmatrix},$$

and as its third component a vector containing the eigenvalues of this matrix. Use your structure to evaluate the m_{34} element of this matrix, and the third eigenvalue (according to the order in which MATLAB computes them).

5b. In order to define *mypoly* as a cell array, we need to know that the basic format for cell arrays is *mypoly{1},mypoly{2},* etc. (note the curved brackets) or more generally *mypoly{index1,index2,etc.}*. Consider the following MATLAB session:

```
>>mypoly{1}='My favorite polynomial';
>>mypoly{2}=@(x)x^2-x-1;
>>mypoly{3}=solve('x^2-x-1');
>>mypoly
mypoly =
'My favorite polynomial' @(x)x^2-x-1 [2x1 sym]
>>mypoly{1}
ans =
My favorite polynomial
>>mypoly{2}(1)
ans =
-1
>>mypoly{3}(1)
ans =
1/2 - 5^(1/2)/2
>>mypoly{3}(2)
ans =
5^(1/2)/2 + 1/2
```

Create a MATLAB cell array that accomplishes the same thing your MATLAB structure accomplished in (a).