Physics 263: MATLAB Cheatsheet II

This is the second collection of basic information and techniques for using MATLAB. These involve both the "symbolic" (`x` is a variable without any particular value) and the "numerical" (`x` is a particular number or vector of numbers or matrix of numbers) functions of MATLAB.

1. **Numerical Two-Dimensional Plotting**

   In "MATLAB Cheatsheet I," plotting symbolic expressions using `ezplot` was discussed. Here we outline how to make two-dimensional graphs using `plot`, `line`, and `loglog`, which take arrays of numerical values as input.

   a. To plot $\sin x$ from $-\pi$ to $+\pi$, we first define a set of points to plot, in the form of a vector (which is considered a $1 \times N$ matrix by MATLAB). We can specify a set of evenly spaced points with spacing 0.1 by:

   ```
   x = -pi:0.1:pi
   ```

   To avoid seeing this vector printed out, add a semicolon to the end:

   ```
   x = -pi:0.1:pi;
   ```

   If instead we want a specified number of points (say 100), then

   ```
   x = linspace(-pi,pi,100);
   ```

   will do ("linspace" is short for "linear space").

   b. Then we give the x vector and the function to plot as the y vector to the `plot` command:

   ```
   plot(x,sin(x))
   ```

   which should pop up a figure window or make a new plot in an existing window (which may be hidden by other windows). This function makes a regular linear-linear plot.

   c. To add a second curve to the same graph (e.g., of $\cos x$ in the same range), use:

   ```
   line(x,cos(x))
   ```

   d. To make a log-log plot instead, we will typically want to space the points *logarithmically* rather than linearly. To set x equal to 150 points from 10^{-5} to 10^{2}, we can use

   ```
   x = logspace(-5,2,150);
   ```

   To plot $x^2 e^{-x}$ from 10^{-10} to 10^{1} with 200 points:

   ```
   x = logspace(-10,1,200);
   loglog(x,x.^2.*exp(-x))
   ```

   Note that we use `.^` and `.*` rather than `^` rather than `*`. The "`.`" means to exponentiate or multiply (or divide, etc.) each term in the vector x, rather than trying to operate on entire vectors or matrices. If you forget the "`.`" you'll get an error message.

2. **Using the Symbolic Toolkit (cont.)**

   Recall that to use variables like x and y and a that are not assigned numerical values, we have to declare them as "symbolic". We'll do that here using `syms`. For example,

```
>> syms x y a
```

delares the symbols x, y, and a, which we can then use to perform integrals, etc.

a. **Evaluating (symbolic) indefinite integrals.** To calculate the integral of e^{-ax} (for example), declare a and x to be symbolic and then use the `int` function:

```
>> syms a x
>> int(exp(-a*x))
```

To specify that $x \sin y$ should be integrated with respect to y (with x treated as a constant):

```
>> syms x y
>> int(x*sin(y),y)
```

(that is, the second argument is the integration variable).

b. **Evaluating definite integrals.** To calculate the integral of e^{-x^2} from 0 to ∞:

```
>> syms  x
>> int(exp(-x^2),0,inf)
```

To specify that $u \sin xu$ should be integrated with respect to u from 0 to $\pi/2$ (with x treated as a constant):

```
>> syms u x
>> int(u*sin(x*u),u,0,pi/2)
```

3. **Numerical Derivatives and Round-off Errors using MATLAB**

In class we considered three numerical approximations to the derivative of $f(x)$ at x_0:

$$\boxed{1.} \qquad f^{(1)}(x_0) \doteq \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x}$$

$$\boxed{2.} \qquad f^{(1)}(x_0) \doteq \frac{f(x_0) - f(x_0 - \Delta x)}{\Delta x}$$

$$\boxed{3.} \qquad f^{(1)}(x_0) \doteq \frac{f(x_0 + \Delta x/2) - f(x_0 - \Delta x/2)}{\Delta x}$$

and found that the third method was much more accurate on a test case than the first two.

The `deriv_test.m` script, which uses the `test_function1.m` function is set up (initially) to try out the first approximation. You can get brief help on these through:

```
>> help deriv_test
>> help test_function1
```

and you run `deriv_test.m` by typing the name (without the `.m` ending) at the command prompt: `>> deriv_test`

The new features in these "M" files are the "switch" construction (which is pretty much self-explanatory) and the plotting, which uses vectors as described in part 1. of this sheet.