Practice 3

In this note, we will practice programming functions in Matlab. There are several ways to define a function in Matlab. We will discuss two common ways: using *script file* and using *function handle*.

1. Let us consider the factorial function. We want to enter a positive integer n to receive the value of $n! = 1.2.3...n$. Matlab has a built-in function to do this, called *factorial*. However, let us make our own factorial function. Call this function '*fac*'. One can do so by writing on a blank script file:

```
function y = fac(n)
s = 1;
for i = 1:n
    s = s*i;
end
y = s;
```

   Then save the file. The name of the file must match the name of the function. Since the function's name is 'fac', the file name must be fac.m. The first line is most important: y is the returning value of the function, 'fac' is the name of the function, n is the argument. In the workspace of Matlab (where you see the prompt '>>'), try

```
>> fac(11)
```

2. A function can call itself or other existing functions in the workspace. For example, the 'fac' function is equivalent to the function 'fac1' defined as follows:

```
function y = fac1(n)
if n == 1
   y = 1;
else
        y = n * fac1(n-1);
end
```

3. The input and output of a function can be arrays/matrices. For example, consider the function $f(x) = \cos x + \sin x$. One can allow x to be an array, i.e. $x = [x_1\ x_2\ \ldots x_n]$, and want the output to be $[y_1\ y_2\ \ldots y_n]$ where $y_i = \cos x_i + \sin x_i$. This function can be programmed as:

```
function y = f(x)
y = cos(x) + sin(x);
```

   Note that the name of this file must be 'f.m'. If one wants to change the function's name, the file name must also be changed accordingly. While coding in Matlab, people tend to avoid using one-lettered names for functions because these names will then be protected and no longer be available for naming variables.

4. Another way to define functions is by using function handle. This method is used when either *the function is quite simple to program* or when one wants to *temporarily rename an existing function without changing its file name*. This is done by using the symbol @. For example, while writing a long program, referring to the name 'fac' all the time may be inconvenient. We want to refer to it as 'g' without changing the function's true name. We write

```
>> g = @fac
```

Now g is the "handle" of function 'fac'. Try the following:

```
g(5)
fac(1) + fac(4) + fac(2) + fac(5) + fac(7) + fac(3) + fac(8)
g(1) + g(4) + g(2) + g(5) + g(7) + g(3) + g(8)
h = @fac
h(5)
```

5. When a function is too simple to program and plays an insignificant role in a procedure, we sometimes don't want to write a separate script file for it. For example, the function $f(x) = \cos x + \sin x$ can be defined in one line as:

```
>> h = @(x) cos(x) + sin(x);
```

Now 'h' is the "handle" of the expression $\cos x + \sin x$. Try

```
h(pi)
h(pi/2)
h([0 pi/2 pi])
h(g(pi))
```