Problem 4.1: Writing a function to perform multiple matrix operations

We have two matrices to be used in the operations \((A, B)\), and four outputs, the results of \(\text{sum}(s)\), \(\text{difference}(d)\), \(\text{element-by-element product}(ep)\), and \(\text{matrix product}(mp)\). The function can be implemented as below.

```matlab
function [s,d,ep,mp]=bop(A,B)
% Problem 4.1: Writing a function to perform multiple matrix operations
% Input arguments: two same size matrices
% Output arguments: sum(s), difference(d), element by element product(ep)
% and matrix product(mp) of two matrices
%
% sum of two matrices
s=A+B;
% difference of two matrices
d=A-B;
% element-by-element product of two matrices
ep=A.*B;
% matrix product of two matrices
mp=A*B;
```

The output of calculating sum, difference, element-by-element multiplication, and matrix
multiplication of of \[\begin{bmatrix} 10 & 35 \\ 70 & 100 \end{bmatrix}\] and \[\begin{bmatrix} 5 & 3 \\ 9 & 25 \end{bmatrix}\] with this function is shown as below.

\[
>>> [s,d,ep,mp]=bop([10 35 ; 70 100],[5 3 ; 9 25])
\]
\[
s = \begin{bmatrix} 15 & 38 \\ 79 & 125 \end{bmatrix}
\]
\[
d = \begin{bmatrix} 5 & 32 \\ 61 & 75 \end{bmatrix}
\]
\[
ep = \begin{bmatrix} 50 & 105 \\ 630 & 2500 \end{bmatrix}
\]
\[
mp = \begin{bmatrix} 365 & 905 \\ 1250 & 2710 \end{bmatrix}
\]

Problem 4.2: Integrating two functions numerically over the interval with function handler

i) With ‘quad’ function, you can obtain numerical integration over [1,10]. First, you should define the function you will evaluate, and calculate numerical value for given integration interval.

\[
>>> P=@(x)x.^2+2*x+3; \quad \% Define Polynomial function given at problem 4.2 i)
\]
\[
>>> quad(P,1,10) \quad \% Integrate function from 1 to 10
\]
\[
ans = 459
\]

ii) The procedure to evaluate Gaussian function is same as i), but we have indefinite integration interval. ‘quad’ function can be used only for the integration over definite interval. However, we can assume that indefinite number is approximated to very large number such as \(10^{10}\), since Gaussian function goes to 0 as \(x\) goes to infinity (or – infinity), and we get very close number to what we get with infinite evaluations.
For i), numerical solution and analytical solution is identical. Due to some limitations of MATLAB number expression, the integration result for i) seems to be 1.0000 which is equal to analytical solution for this Gaussian function, but actually 1.0000123047690…(with more significant digits) The reason why they are different is that we just evaluate function up to certain finite number, and there always exists error between numerical value and true value. Therefore, evaluating tolerance is quite important in the numerical analysis. This evaluation is acceptable if we have tolerance of $10^{-5}$, but it’s not if we have $10^{-6}$.

**Problem 4.3 : Calculating the factorial of non-negative integer**

The factorial of integer $n$, $n!$, is defined as below:

$$ n! = n \times (n-1) \times (n-2) \times \cdots \times 2 \times 1 $$

$$ 0! = 1 $$

Therefore, $n!$ can be calculated with ‘for’ loop by multiplying loop counter to the output. However, special case of $n=0$ should be considered. Input argument ($n$) is non-negative integer, and output argument ($o$) is the factorial of input argument.

```matlab
function o=fctrl(n)

% Problem 4.3: Calculating the factorial of non-negative integer with function

% Define 0!=1
o=1;
```
% Calculate factorial
% if n<1, MATLAB skips 'for' loop (for n=0)
for i=1:n
    o=o*i; % n!=(n-1)!*n
end

The output of 125! will be displayed as below.

>> fctrl(125)     % Calculate 125!
ans =
1.8827e+209