Module 1 Practice Problems (Solutions)

Loops, Arrays, and Vectorization

1. Only part (a) is explained, the others are done in the same fashion.

   (a) Let $x = 5.4$ and $s = \cos(x) = \cos(5.4)$. Denote by s_k the kth partial sum of the Taylor series given,

   $$s_k = \sum_{n=0}^{k} \frac{(-1)^n x^{2n}}{(2n)!} = \sum_{n=0}^{k} \frac{(-1)^n (5.4)^{2n}}{(2n)!},$$

   and so

   $$s = \lim_{k\to\infty} s_k.$$

   We calculate $s_0, s_1, \ldots, s_N$ using a for-loop:

```
N = 10; x = 5.4; s = cos(x);
sk = 0;
for n = 0:N
    sk = sk + (-1)^n * x^(2*n) / factorial(2*n);
    fprintf('%3d %22.16f %24.16e\n', n, sk, abs(sk-s))
end
```

2. Recall that a vector, by default, is a column vector in this class; this is why each of the following answers has the transpose `.'`. If a row vector is desired, it will be explicitly mentioned.

   - **LM** 3.1–3(b): Note that $b_k = (2k-1)^2$ for $k = 1, 2, \ldots, p$ where p is the number of elements of $\mathbf{b}$, which is to be determined. Since the elements are all required to be $\leqslant n^2$, it must be that p is the largest positive integer such that $2p - 1 \leqslant n$. This implies that $p = \lfloor (n+1)/2 \rfloor$ (floor function). However, as far as generating the vector on MATLAB is concerned, we can simply do

```
b = ([1:2:n].') .^ 2;
```

     without needing to type p out explicitly. (Think about why and make sure you understand why.)

   - **LM** 3.1–3(c): In this part, $c_k = (2k-1)^2$ for $k = 1, 2, \ldots, q$ where q is the number of elements of $\mathbf{c}$. The requirement is that the elements are $\leqslant n$ not n^2. Thus q is the largest integer such that $(2q-1)^2 \leqslant n$, *i.e.*, $q = \lfloor \left(\sqrt{n} + 1\right)/2 \rfloor$. Though the expression for q is messy, it can be compactly coded as

```
c = ([1:2:sqrt(n)].') .^ 2;
```

     since the colon operator ensures that the last elements of `1:2:sqrt(n)` does not exceed $\sqrt{n}$.

- **LM** 3.1–3(e):

```
        e = [2:n^2, 999999].';
```

It is important that you do not omit `.'` at the end since $\mathbf{e} = (2, 3, \ldots, n^2, 999999)^{\mathrm{T}}$, a column vector!

- **LM** 3.1–3(g): All angles are in radians.

```
        g = [sin(1:n), cos(n:-1:1)].';
```

- **LM** 3.1–4(c): The terms can be viewed as

$$
\begin{aligned}
t_1 &= 1^2 + 1 = 2 \\
t_2 &= 2^2 + 1 = 5 \\
t_3 &= 3^2 + 1 = 10 \\
&\vdots \\
t_j &= j^2 + 1, \quad \text{for } j \geqslant 1.
\end{aligned}
$$

Therefore,

```
t = ([1:n].^n + 1).';
```

- **LM** 3.1–4(e): Note that the elements of $\mathbf{v}$ are powers of 2:

$$2^{-1}, 2^0, 2^1, \cdots, 2^{n-1}.$$

The last term is 2^{n-1} since n terms are needed. Hence,

```
v = 2 .^ ([-1:n-2].');
```

- **LM** 3.1–5(d):

```
d = ( 1 ./ sin(n:-1:1).^3 ).';
```

- **LM** 3.1–5(f):

```
f = factorial(1:n+1).';
```

- **LM** 3.1–16:

```
s = A(1,:) + A(2,:);        % sum of first two rows
A(2,:) * A(3,:)';           % inner product
B = A(:,3) * A(:,7)';       % outer product
C = A(4,:)' * A(:,9)';      % outer product
d = diag(A);                % diag can take a rectangular matrix
e = [diag(A); zeros(10,1); % vertical concatenation
```

- **LM** 3.2–7: As per the instruction found at the beginning of the exercises, we answer this question without using loops. For part (a), first make a copy of `A` and call it `B`. Then find where `B` is positive using the `find` function and change the corresponding elements of `B` to zeros.

```
B = A;
B(find(B<0)) = 0;
```

Alternately, logical arrays can be used instead of `find` as shown below.

```
B = A;
B(B<0) = 0;
```

Part (b) is done similarly.

```
C = A^2 - 100;
C(find(C<0)) = 0;
```

Note that A is a square matrix and the expression A^2 is equivalent to $A \cdot A$, the multiplication of two matrices learned in linear algebra, not an elementwise multiplication.

For part (c), use the `max` function:

```
D = max( A-10, B-100 );
```

3. (a) Already given as a hint.

   (b) Similar to part (a).

```
p = 1;
for j = 1:n
    p = p * x(j);
end
fprintf('Result using ''prod'' function: %24.16f\n', prod(x));
fprintf('Result using a ''for'' loop  : %24.16f\n', p);
```

   (c) See p. 91 of Module 1 lecture slides.

   (d) Same as above.

   (e) The given expression is the average or the *mean* of the elements of **x**. It is computed using a loop similar to the one for part (a) because of the obvious reason. To vectorize, use can either do `sum(x)/n` or `mean(x)`.

   (f) See Problem 1.

   (g) Modify the previous part.

4. (a) **x** consists of 11 equispaced points on $[0, 2\pi]$, including both endpoints. So use `linspace`. Assume that `n` is already stored.

```
x = linspace(0, 2*pi, n);
y = sin(x);
```

   (b) Assume again that `n` is already stored. Using the `cumsum` function:

```
y = cumsum(1:n);
fprintf('Sum of integers 1 to %2d: %5d\n', [1:n; y]);
```

More solutions to be added.