HW03 Hints

1. (Array construction) Use ONE MATLAB statement to generate each of the following arrays, where you can assume that a positive integer n is already stored in MATLAB. We are only interested in MATLAB statements, and you will be graded on the correctness of your code alone. Do NOT show any outputs.

   (a) The column vector $\mathbf{a}$ where $a_1 = 1$, $a_2 = 3$, $a_3 = 5$, etc., as long as the elements are $\leq n$.

   (b) $\mathbf{b} = \left(2, 3, 4, 5, \ldots, n^2, 999999\right)^{\mathrm{T}}$.

   (c) $\mathbf{c} = (\sin 2, \sin 5, \sin 8, \ldots, \sin(-1 + 3n))^{\mathrm{T}}$.

   (d) $\mathbf{d} = \left(\frac{1}{2}, 1, 2, 4, 8, 16, \ldots\right)^{\mathrm{T}} \in \mathbb{R}^n$.

   (e) $\mathbf{e} = \left(2^{-n}, 2^{-(n-1)}, 2^{-(n-2)}, \ldots, 2^{n-2}, 2^{n-1}, 2^n\right)^{\mathrm{T}}$.

   (f) $\mathbf{f} = (0, 2, 4, 1, 3, 0, 2, 4, 1, 3, \ldots)^{\mathrm{T}} \in \mathbb{R}^{5n}$ using the `mod` function, and no other MATLAB function.

   (g) $A \in \mathbb{R}^{m \times n}$, where $A_{i,j} = n(i-1) + j$ for $1 \leq i \leq m$ and $1 \leq j \leq n$, i.e.,

   $$A = \begin{bmatrix} 1 & 2 & 3 & \cdots & n \\ n+1 & n+2 & n+3 & \cdots & 2n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (m-1)n+1 & (m-1)n+2 & (m-1)n+3 & \cdots & mn \end{bmatrix},$$

   using the `reshape` function.

   (h) The $n \times 3$ matrix

   $$B = \begin{bmatrix} 1 & 1^2 & 1^3 \\ 2 & 2^2 & 2^3 \\ 3 & 3^2 & 3^3 \\ \vdots & \vdots & \vdots \\ n & n^2 & n^3 \end{bmatrix}.$$

   Note 1. Each answer must be **ONE** MATLAB statement, not one line, written in terms of `n`.

   Note 2. One suggested workflow is to insert

   ```
   n = 17;   % or some other manageable number
   ```

   at the beginning of the problem, and run your code so that you are confident it works correctly. Once you are confident, suppress your outputs by putting a semicolon at the end of each statement.

2. (nth roots: adapted from **LM** 3.1–24) Recall that

$$1 = e^{0\pi i} = e^{2\pi i} = e^{4\pi i} = \cdots ,$$

but, if we take the nth root, we obtain n distinct roots of $x^n - 1 = 0$:

$$e^{0\pi i/n}, e^{2\pi i/n}, e^{4\pi i/n}, \ldots .$$

These n distinct complex numbers are called the nth *roots of unity*. One can express all solutions of $x^n - a = 0$, where $a > 0$, in terms of the roots of unity:

$$\sqrt[n]{a} e^{0\pi i/n}, \sqrt[n]{a} e^{2\pi i/n}, \sqrt[n]{a} e^{4\pi i/n}, \ldots .$$

   (a) Write a script which, given a positive integer n and a positive real number a, finds all n roots of $x^n - a$ at once, using ONE statement. It must also print out all n of these roots neatly using either `disp` or `fprintf` in tabular form. A loop may be used for printing results, but is not allowed in the calculation of the roots.

   (b) Run the script with $n = 3, 5, 7$, and 11.

   Note. Print out the contents of your script m-file using `type`.

3. (Strange behavior of a continuous function: adapted from **LM** 3.1–25) Consider the function

$$g(x) = \begin{cases} \dfrac{\log(1+x)}{x} & \text{if } x > -1 \text{ and } x \neq 0 \\ 1 & \text{if } x = 0, \end{cases}$$

This is a continuous function for all $x > -1$, but numerically it has difficulties when $x \approx 0$.

   (a) Check this yourself by letting $x = 10^{-k}$ for $k = 1, 2, \ldots, 20$. Generate a nice table. Do it in a single code block; you do not need to write a script for this. Do it without using any loop.

   (b) Repeat the previous part but rewrite $g(x)$ as

$$g_2(x) = \begin{cases} \dfrac{\log(1+x)}{(1+x) - 1} & \text{if } x \neq 0 \\ 1 & \text{if } x = 0, \end{cases} .$$

   Calculate the denominator as written — **do not convert it to** x. Do it without using any loop.

   (c) Summarize the results of the two parts into a single table. The table should have three columns with the first being x, the second being $g(x)$, and the last being $g_2(x)$. Use `format long g` for your output if using `disp`; use a compatible format specification if using `fprintf`. Do it without using any loop.

   Note. Write your code inside a single code block for each part. You do not need to write a script for this problem.

4. (Data manipulation exercise – analyzing grades) Download `grades.dat` and load it using

```
>> grades = load('grades.dat');
```

To see how the data are organized in the data file, use `type grades.dat`. Minimize the use of loops and conditional statements as much as possible.

(a) Determine the number of students.

(b) Compute the total grade according to the weights specified in the header. Do this without using a loop.

(c) The letter grades are determined by

- A: $[90, 100]$
- B: $[80, 90)$
- C: $[70, 80)$
- D: $[60, 70)$
- E: $[0, 60)$

Find the number of students earning each of the letter grades.

5. (Approximating π, another take: **LM** 3.2–11) There are many infinite sums which involve π, such as

$$\sum_{k=1}^{\infty} \frac{\sin k}{k} = \frac{\pi - 1}{2}$$
$$\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6}$$
$$\sum_{k=1}^{\infty} \frac{1}{k^4} = \frac{\pi^4}{90}.$$

(a) Calculate each sum as accurately as you can by letting $\sum_{k=1}^{\infty} \to \sum_{k=1}^{n}$. Tabulate the error for $n = 10^{\ell}$ where $\ell = 2, 4, 6$, and 8 if possible; otherwise, end with $\ell = 7$. Do this by creating a vector which contains all the elements, e.g., $\mathbf{v}_2 = (1, 1/2^2, 1/3^2, \ldots, 1/n^2)$, and then using `sum` to calculate the sum of all the elements.

(b) Repeat the previous part but add the terms from smallest to largest. That is, instead of calculating the terms by

```
k_vec = [1:n];
```

use

```
k_vec = [n:-1:1];
```

Are any of the answers more accurate?