Updated on January 28, 2022

Math 3607: Homework 3

Wednesday, February 2, 2021

TOTAL: 30 points

- Problems marked with are to be done by hand; those marked with are to be solved using a computer.
- **Important note.** Do not use *Symbolic Math Toolbox*. Any work done using `sym` or `syms` will receive NO credit.

1. (Array construction: selected problems from **LM** 3.1) 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 $a_1 = 1^2$, $a_2 = 3^2$, $a_3 = 5^2$, etc., as long as the elements are $\leq n^2$.

   (b) $\mathbf{b} = \left(\frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots, \frac{1}{2n}\right)^{\mathrm{T}}$.

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

   (d) $\mathbf{d} = (2, 5, 10, 17, 26, 37, \ldots)^{\mathrm{T}} \in \mathbb{R}^n$.

   (e) $\mathbf{e} = \left(1^1, 2^{1/2}, 3^{1/3}, \ldots, n^{1/n}\right)^{\mathrm{T}}$.

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

   (g) The $n \times n$ matrix A, where $A_{i,j} = (n(i-1)+j)^2$ for $1 \leq i, j \leq n$, i.e.,

   $$A = \begin{bmatrix} 1 & 4 & \cdots & n^2 \\ (n+1)^2 & (n+2)^2 & \cdots & (2n)^2 \\ \vdots & \vdots & \ddots & \vdots \\ (n^2-n+1)^2 & (n^2-n+2)^2 & \cdots & n^4 \end{bmatrix},$$

   using the `reshape` function.

   (h) The $(n+1) \times 4$ matrix

   $$B = \begin{bmatrix} 1 & 0 & 0^2 & 0^3 \\ 1 & 1 & 1^2 & 1^3 \\ 1 & 2 & 2^2 & 2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & n & n^2 & n^3 \end{bmatrix}.$$

   Note 1. Each answer must be **ONE** MATLAB statement, not one line.

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. (Roots of unity revisited: adapted from **LM** 3.1–24) In Problem 2, Homework 1, we calculated all five roots of $x^5 + 1$, but it required a number of lines of code. In this problem, we will do this more compactly for a more general $x^n + 1$. Recall that

$$-1 = e^{\pi i} = e^{3\pi i} = e^{5\pi i} = \cdots ,$$

but, if we take the nth root, we obtain n distinct values

$$e^{\pi i/n}, e^{3\pi i/n}, e^{5\pi i/n}, \ldots,$$

due to the periodicity of the complex exponentials; think Euler.

   (a) Write a script which, given a positive integer n, finds all n roots of $x^n + 1$ 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: **LM** 3.1–25) Consider the function

$$f(x) = \begin{cases} \dfrac{e^x - 1}{x} & \text{if } x \neq 0 \\ 1 & \text{if } x = 0, \end{cases}$$

This is a continuous function for all $x \in \mathbb{R}$, but numerically it has difficulties when $x \approx 0$.

   (a) Check this yourself by letting $x = 2^{-k}$ for $k = 1, 2, \ldots, 50$. Generate a nice table. Do it without using any loop.

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

$$f_2(x) = \begin{cases} \dfrac{e^x - 1}{\log e^x} & \text{if } x \neq 0 \\ 1 & \text{if } x = 0, \end{cases}$$

   which was proposed by Bill Kahan. Leave the denominator as $\log e^x$ — **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 $f(x)$, and the last being $f_2(x)$. 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. (Vectorized data manipulation: from Su21 midterm) You are planning a bicycle trip along a 400 mile stretch of a very straight midwestern rural highway and plan to stop each night at a different town. The towns are irregularly spaced, but you have mileage markers for each town, given in the array

$$\texttt{Miles} = [0, 27, 69, 101, 120, 154, 178, 211, 235, 278, 306, 327, 356, 391, 400]$$

You would like to compute the distances you have to travel each day, as well as other statistics about your trip.

First, create an array `Miles` containing the mile markers above. Then answer the following questions without using a loop; a correct answer obtained using a loop will earn half of the allotted points.

(a) Create an array `Dist` containing the distances between each of the mile markers.

(b) What is the shortest distance you will have to bicycle on any day?

(c) What is the longest distance?

(d) What is your average daily distance?

(e) How far will you have to go on day 7?

(f) If you would like to stop each day for lunch at exactly the halfway point for each day's journey, what mileage values should you plug into your GPS to assure that you do not miss lunch? These mileages values are distances from the very beginning of the trip, not from the beginning of each day.

(g) How can you recover your array `Miles` from your array `Dist`?

Hint. Make use of the functions listed on p. 11 of Lecture 7 slides.

Note. Write one MATLAB statement for each part in a single code block. You do not need to write a script for this problem.

5. (Birthday problem: last exercise, Lecture 7) In a group of n randomly chosen people, what is the probability that everyone has a different birthday?

(a) Find this probability by hand. The answer must be written in terms of n.

(b) Let $n = 30$. Write a script that generates a group of n people randomly and determines if there are any matches. No loops nor if-statements are allowed in the script.

(c) Write another script by modifying the previous one to run a number of simulations and numerically calculate the probability. This script should take the number of simulations as an input. No loops nor if-statements are allowed in the script.

(d) Run the script with $1000, 10000$, and 100000 simulations. Compare the result with the analytical calculation done in part (a).

Note. For parts (b) and (c), print out the contents of your script m-files using `type`.