Matlab Homework 9*a*

In the online book:

- Do the "Challenge Activities" of: 11.1-3; 12.1,3,4,
- Do the "Participation Activities" of: 9.1,2, 12.6

Matlab Homework 9*b*

The same general requirements as for homework 4b apply. And you must study the posted lesson(s) and have done the online book part above before you can ask a TA or the instructor for help.

1. In an earlier homework, (homework 3?), you printed out the roots ω_1, ω_2, ω_3, and ω_4, of the following equation:

   $$J_0(\omega) - k\omega J_1(\omega) = 0$$

   where J_0 and J_1 were Bessel functions of the first kind and the given constant k was a nondimensionalized flexibility of the membrane attachment.

   Repeat this, but now no longer use separate code for each frequency that you print out. Instead use a `for` loop over counter n, going from 1 to value $n_{\max} = 9$ (instead of 4), to find and print the first 9 roots. (So you must use one piece of code, not 9, for the 9 frequencies.) Take $k = 2$ again.

   The correct *old* solution is already in the `q1.m` file. Just modify it as requested.

   Octave users: The Octave `fzero` does not always find the closest root for some reason. Just live with it. The interval method, which is safe, will work the same as in Matlab.

2. In an earlier homework, (homework 8?), you created the matrix

   $$A = \begin{pmatrix} -2 & 1 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 1 & -2 \end{pmatrix}$$

   by typing in all those numbers. This is typo-prone and you have to do it all again with a vengeance when you want to try much more points.

   While in a later question in that homework, you created the matrix using the Matlab `diag` and `ones` functions, the possibility of of doing that is very limited. And it is very difficult to understand for someone reading your code.

   So, in this question create a script `vibMatrix` that, given a value for variable n, creates an $n \times n$ matrix of the above type by initializing it to zero and then using a `for` loop to put the various nonzero elements in it. Run the script for n equal to 4, 5, and 6, and check that in each case, you get the right matrix. (Do not use any `if` statements; since the first and last rows are not like the rest, do them separately outside the `for` loop.)

3. Some mathematician claims that the sum

$$\sum_{i=1}^{i_{\max}} \frac{1}{i}$$

in other words

$$\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \ldots + \frac{1}{i_{\max}}$$

becomes infinite when $i_{\max}$ becomes infinite. Let's check that out.

In a script `sum1.m`, create Matlab code that performs the sum up to a given $i_{\max}$. Check the script by first taking $i_{\max}$ to be 10; you should get about 2.929.

Next put that summation script inside an "outer" `for` loop on $i_{\max}$ where you take $i_{\max}$ to be the successive values [10, 100, 1000, 10000, 100000]. This loop should be in your `q3.m` script itself.

After studying the values you get for the sum using these five $i_{\max}$ values, use `disp` to comment on whether it looks like the sum converges to a definite value when $i_{\max}$ becomes bigger and bigger, or whether it looks like the value seems to keep getting bigger and bigger.

Warning: Summing 100,000 terms may be a bit slow on some computers. You may want to wait with that one until everything works OK. Initially just do [10, 100, 1000, 10000].

4. The following function, the "sine integral",

$$\mathrm{Si}(x) = \int_0^x \frac{\sin \xi}{\xi} \, \mathrm{d}\xi$$

cannot be expressed in terms of simpler functions. You cannot find the needed antiderivative in terms of normal functions, even though the integrand *seems* so simple.

However, the Taylor series of the Si function is easy to find. (Just write the Taylor series for $\sin \xi$, divide by ξ, and integrate.) The result is

$$\mathrm{Si}(x) = \frac{x}{1!1} - \frac{x^3}{3!3} + \frac{x^5}{5!5} - \frac{x^7}{7!7} + \ldots$$

which can be written as

$$\mathrm{Si}(x) = \sum_{\substack{i=1 \\ i \text{ odd}}}^{i_{\max}} (-1)^{(i-1)/2} \frac{x^i}{i!i}$$

Write a script `si.m` (lowercase) that, given x and $i_{\max}$, sums this sum. Note: You can skip the even values of i in the `for` loop using a START:STEP:END construct in the `for` command for a suitable value of STEP. (STEP is the difference between successive i-values.)

For reasons to be explained later, initialize the sum to the term $i = 1$, (i.e. `total=x`), then start the `for` loop at $i = 3$ to add the other terms.

Use the script to compute $\mathrm{Si}(5\pi)$. The correct value is about 1.6339648 according to my table book. Experiment with the minimum value of $i_{\max}$ you need to get the value correct to the given number of digits. Print out as

```
Value: 1.12345678
Table: 1.6339648
I needed to sum * terms to get this.
```

Use `fprintf` commands to do so, without data numbers in the FORMATSTRING.

Note: Si is actually a quite important function, and Matlab provides this function as "sinint". (It is apparently within the symbolic logic package; at least in Octave it is.) I would not be surprised if they used some adulterated Taylor series to evaluate the function for relatively small values of x.