Matlab Homework 2c

The same requirements as for homework 1c apply. In addition, no longer use the default variable 'ans'. Store any number that the grader needs to see in a variable with an easily understandable name.

Do not print out hundreds of plot point values. Your plots should show such numbers graphically.

Motivation: Consider a drum whose membrane is flexibly attached to the drum rim. If you hit such a drum in the center of the membrane, the nondimensionalized frequencies (tones) ω that are produced satisfy the equation

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

where J_0 and J_1 are Bessel functions of the first kind and the given constant k is a nondimensionalized flexibility of the membrane attachment. In the current homework, assume that $k = \frac{1}{2}$.

1. Plot the two functions $J_0(\omega)$ and $k\omega J_1(\omega)$ together in a single graph. Take $k = \frac{1}{2}$. Take the horizontal axis to extend from 0 to 3.5π, with labeled tick marks at whole multiples of π. Take the vertical axis big enough that the functions do not get cut off, but not much bigger than that. Use a grid to allow you to ballpark the correct frequencies (where the curves intersect). Use about 200 ω values to plot. Use an appropriate xlabel and title.

   Warning: $k\omega J_1(\omega)$ needs to be written as `k*omega.*bess...`; the point before the second star is necessary to tell Matlab that the ω values and the J_1 values are sets of numbers, not vectors. Without the point, Matlab would try to take a *dot product* of the two "vectors" (and fail). See section 5.1 in the online book.

2. Create a function `drumFreqEq0pt5` that gives the error in the equation above for the frequency for a given value of ω, call it `omega`. In this function, assume that $k = 0.5$.

   This function must be in a separate file named 'drumFreqEq0pt5.m'. That is just like `freqEq1` was in file 'freqEq1.m' in the lesson2 example. See also section 3.3 in the book.

   Warning: Do *not*, repeat *not*, give a value to input argument `omega` inside 'drumFreqEq0pt5.m'. Input arguments get values only when the function is *used*, not when it is *defined*.

   Your function must be well commented; compare the posted lecture notes of lesson 2.

   In your 'hw2.m' file, issue a `help drumFreqEq0pt5` command, and verify that that tells you what this function does, what its input arguments are, and what it returns to whoever or whatever uses it. Your answer to question 2 in 'hw2.m' should be of the form

```
%% Question 2
%
% Create function drumFreqEq0pt5.
%
% <include>drumFreqEq0pt5.m</include>

% show that help works
help drumFreqEq0pt5
```

The 'include' tags above adds the contents of 'drumFreqEq0pt.m' to 'hw2.pdf' so that the grader can see the function you created.

3. Ballpark the lowest correct frequency ω_1 by looking at the plot. Take it to be a fraction of π and call it `ballpark`. Use `ballpark` as an initial guess in `fzero` to find the exact lowest frequency ω_1, to be stored in variable `omega1`.

   However, a ballpark can fail. Therefore, *next* find an interval $[a, b]$ that is big enough that you can be sure that the first frequency is inside, but not so big that more than one frequency is inside. You can find a and b values like that that are whole multiples of π, so do so. Make sure that Matlab prints out the values of a and b that you selected in the published version (like in `a=...`, `b=...`). Check then that `drumFreqEq0pt5` is of different sign at the end points a and b. Use variable names `errora` and `errorb`. Then check that using this interval, you get the same answer with `fzero` as before or fix it. The interval method never fails if used correctly.

4. Looking at a mathematical handbook, it can be shown that for high enough frequencies, the frequencies should by approximation be given by

$$\omega_n \approx (n - \tfrac{3}{4})\pi$$

   Use these approximate values as initial guesses in `fzero` to find the accurate values for ω_n for $n = 1$ to 4. Matlab should print them out like `omega1=...`, `omega2=...`, ... (You will want to use copy and paste. Later on we will cover `for` loops, which will make this much neater.)