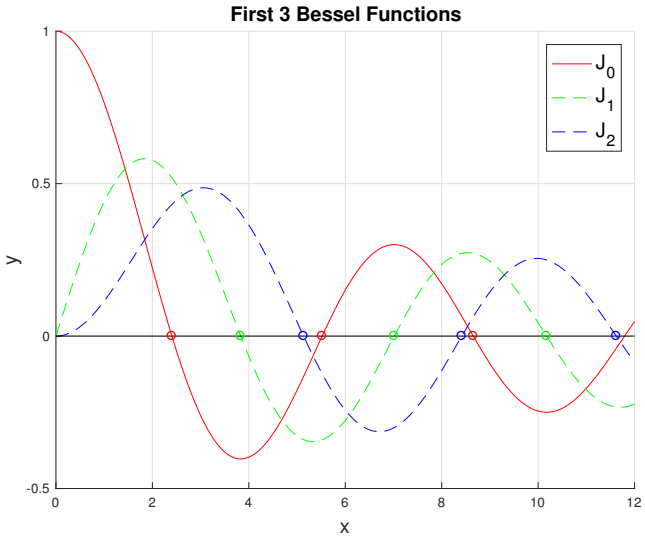
Crowdmark Due Date: Monday 11 February, 2019 at 11:55pm

- Please follow the *Guidelines for Computing Assignments* found on the Canvas FAQ.
- Submit a one-page report via Crowdmark, grading scheme is also found on the FAQ.
- Acknowledge any and all collaborations and assistance from classmates/TAs/instructor.

Finding the Roots of Bessel Functions

The so-called *Bessel Functions*, denoted as $J_m(x)$ where $m = 0, 1, 2, ...$ is an integer, are a class of special functions with many applications in mathematics and science. A plot of the first three Bessel Functions can be found in figure 1. In this figure we have marked the first three roots of each, and in general we denote the k^{th} **positive** root of $J_m(x)$ as $z_{m,k}$. However, since $J_m(x)$ has no closed form representation for any m, there is no way to find $z_{m,k}$ analytically for general m and k. **Your mission, as you shall do in this assignment, is to find the roots $z_{m,k}$ using both Newton's method and Matlab's *fzero*, as the basis for comparing the two in terms of accuracy, efficiency, and robustness.** Ultimately we will apply our root-finding efforts to visualize solutions to the *wave equation*, a physically important PDE.



Despite the fact that the $J_m(x)$ are challenging to work with analytically, we do know some special properties that can help us. First and foremost, we can evaluate $J_m(x)$ numerically in Matlab, for any m and x, using the function call *besselj(m,x)*. Second, we have the following three *location properties* about the zeros $z_{m,k}$ that you should in your numerical search:

- A guess for the 1^{st} zero of $J_0(x)$ is $z_{0,1} \approx 2.5$.
- The k^{th} root of $J_m(x)$ is larger than the $(k-1)^{th}$ root by a value that is around π
- The 1^{st} root of $J_m(x)$ is bracketted by the first two roots of $J_{m-1}(x)$; that is $z_{m-1,1} < z_{m,1} < z_{m-1,2}$.

The other piece of information that we need is how to compute derivatives for the Bessel Functions, in particular for implementing Newton's method. There is an incredibly useful recursive relationship for this:

$$J'_m(x) = \frac{1}{2}(J_{m-1}(x) - J_{m+1}(x))$$

Because we can evaluate any of the Bessel Functions numerically at any time using *besselj*, we have an immediate expression for the derivatives.

Your goal in this computation will be to produce a matrix whose entries are roots of the Bessel functions

$$[A_{M,K}] = \begin{bmatrix} z_{0,1} & z_{0,2} & \cdots & z_{0,K} \\ z_{1,1} & z_{1,2} & \cdots & z_{1,K} \\ \vdots & & & \vdots \\ z_{M,1} & z_{M,2} & \cdots & z_{M,K} \end{bmatrix}$$

where $J_m(z_{m,k}) = 0$ (with $k = 1, 2, \cdots, K$) over the range of the Bessel Functions $\{J_0(x), \cdots, J_M(x)\}$. What do you do with this matrix? We have supplied you with a movie-making script that needs $[A_{16,16}]$ as its input. If you run the script without modification, it produces a nonsense surface plot and an animated .gif file. But when you have your matrix of Bessel zeros properly calculated, the script *CA4_BesselMovie.m*, then rewards you with a numerical movie that looks a lot better! You **do not** need to know what this script is doing, other than it takes your input matrix $[A_{16,16}]$ and produces graphical plots.

To complete this assignment, you should do the following:

- Download the script *CA4_bessel.m* from Canvas. This code is already set up to find entries of the matrix $[A_{2,2}]$ using both *fzero* AND Newton's method. This code also tracks the number of function evaluations needed for both methods. However, the initial conditions for Newton's method have **not** been chosen intelligently, so it fails to properly find the roots.
- Your first goal should be to generate 2 codes: one that uses only *fzero* to compute $[A_{2,2}]$, and the other uses only Newton's method. You should use bracketting initial guesses for the *fzero* code. **The three location properties from the first page will help with finding initial guesses/brackets that lead to convergence.**
- Next, we want to introduce some *for* loops into our codes to find $A_{16,16}$
- When you have script working well, produce $[A_{16,16}]$, and modify the *CA4_BesselMovie.m* by removing the indicated *clear* command at the start. It should then produce a moving surface, but whose outer circular edge is stationary with the values $z = 0$ (blue circle, not moving). If this blue circle is moving, then you have some wrong entries. Also, there is a plot (figure 3000) of the outer values to check how close they really are to being zero — these are related to your root-finding tolerance.
- Once you can correctly produce the matrix $[A_{16,16}]$, you can start comparing how the convergence tolerance and choice of initial guesses/brackets effects performance of each method. You should think about **efficiency** in terms of how many function evaluations are required by each method, **accuracy** in terms of how close the boundary values are to numerical zero (figure 3000), and **robustness**. For this last point, recall that the *CA4_Bessel.m* was initialized with values that cause failures.
- Your final report must include your discussion of experimental parameters and results. As well, you need to include a figure of the final frame of the movie generated by *CA4_BesselMovie.m*, using your computed $[A_{16,16}]$.

Optional Reading: If you are interested in learning more about the wave equation PDE being solved in *CA4_BesselMovie.m*, we will direct you to the Wikipedia page for *Vibrations of Circular Membranes*: `https://en.wikipedia.org/wiki/Vibrations_of_a_circular_membrane`

Recall from lecture that you should aim to communicate critical ideas with clarity. To this end, please underline the **three** sentences in your report that you feel convey the biggest impact to the reader.