## **Computing Assignment 4**

## **Finding the Roots of Bessel Functions**

The goal of this assignment is to compare Matlab’s *fzero* function and Newton’s method when finding the roots of the Bessel Functions - Jm(x).

**Matlab’s fzero and Newton’s method**

For both *fzero* and Newton’s method, the initial parameters are essential for convergence. Each iteration of *fzero* requires an interval (a, b) in which to search for the root. This interval must bracket the root otherwise unwanted results will occur. Similarly, Newton’s method requires an initial guess (X0), which is close to the desired root. Using known properties of the Bessel Functions, the initial parameters can be chosen intelligently.

The first zero of J0(x) is known to be around 2.5. The initial interval a - b for *fzero* must bracket this route, therefore, let

[a, b] = [2.4, 2.6].

Note that both a and b are close to 2.5, this is to ensure that no other roots of J0(x) are bracketed. For example, the second root of J0(x) is around 3.8, so setting [2, 4] as our interval would be a poor choice since it brackets two roots of J0(x). For Netwon’s method the initial guess should be close to the root, therefore, let

X0 = 2.5

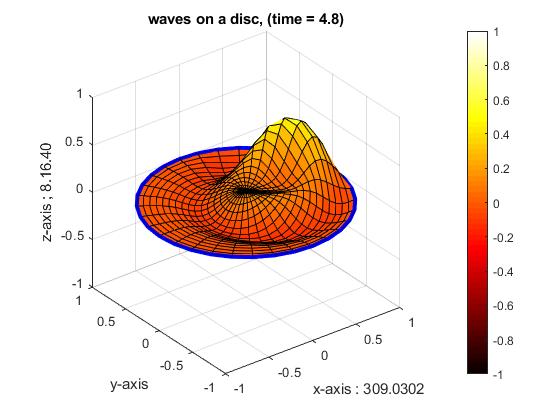
For each root of the Bessel Function Jm(x), the initial bracket/guess gets updated according to the following criteria: the kth root of Jm(x) is larger than the previous root by around π:

kth root = (k-1)th root + π.

For *fzero* our kth interval must bracket the kth root so let

a = kth root – offset  
b = kth root + offset,

where offset is a value of π/2. For Newton’s method use the kth root estimate as the initial value for the kth interval.

For each Bessel Function Jm(x) - excluding J0(x) - the first bracket/guess gets updated according to the following criteria: the first root of Jm(x) is bracketed around the first two roots of Jm-1(x). 

**Obtaining A Smooth Computed Contour**