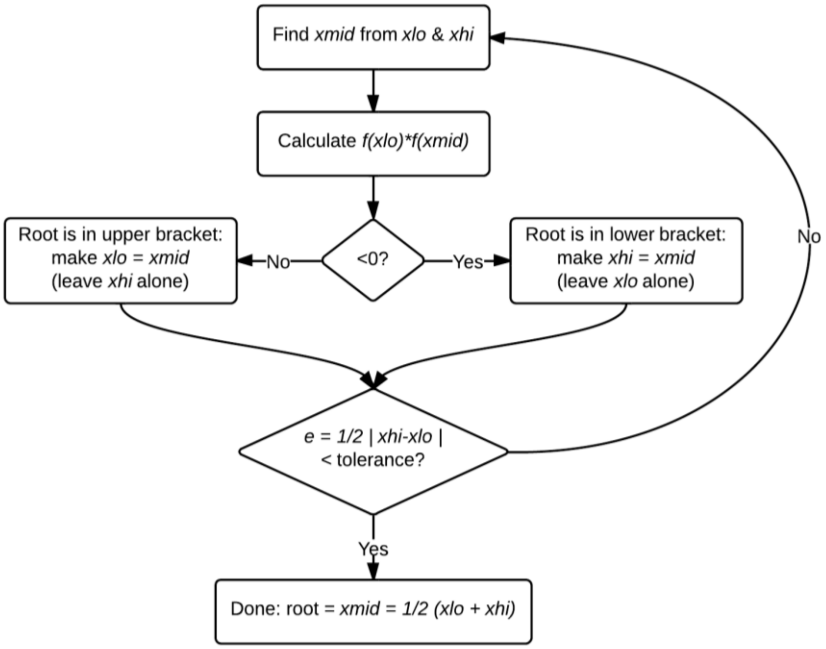
MECE-301 Engineering Applications Laboratory
Assignment #3

The bisection method is a numerical technique for finding the roots of a function *f(x)*. It is a bracketing approach, which means that it will always work provided it is properly configured. It works by establishing two bounds around a chosen root, which we will call *xlo* and *xhi*, for which one of the two function values *f(xlo)* and *f(xhi)* will be positive and one will be negative (again, if properly configured). Let the midpoint of these two bounds be *xmid*, which is simply $\frac{1}{2}(xlow + xhi)$. The value of *f(xmid)* may be positive or negative, depending upon where it lies with respect to the other two bounds and the root itself. The bisection method then works by sequentially moving the two bounds together according to an algorithm, eventually "squeezing" around the root, and making the value of *xmid* converge upon the root. The algorithm progresses until the two bounds are within a specified distance apart, which therefore also bounds the accuracy of the calculated root. Let the tolerance of the calculated root be defined as *e*, where $e = \frac{1}{2}\,|xhi - xlow|$.

The algorithm proceeds as follows: we take the current values of *xlow* and *xhi* to find *xmid*. From these, find *f(xlo)*, *f(xmid)*, and *f(xhi)*. Then we test to see in which "bracket" the root lies: either between *xlo* and *xmid*, or between *xmid* and *xhi*. To do this, we simply test whether the product of the corresponding function values is positive or negative on each (whichever is negative must contain the root, since the function must cross the x-axis there, unless an error has occurred in the original configuration). Finally, we "re-bracket" accordingly, and repeat the process until we have meet a desired tolerance condition. A basic flowchart is shown below (only of the main algorithm, not the entire program).

With this information, write a LabVIEW VI which implements the bisection method to find any desired roots of the function $f(x) = x\,ln(x)\,sin(x^2)$ to a specified tolerance (only one root is determined at a time). The starting values for *xlo* and *xhi* are to be accessed as controls on the front panel, as well as the desired tolerance. Obviously the calculated root (the last value for *xmid*) must also be displayed as an indicator on the front panel. Include a basic error check feature that will terminate the program if the bounds do not contain the root (for whatever reason). The basic algorithm is obviously implemented with a while loop, since it is controlled according to the tolerance specified by the user, and the width of the overall bracket between the lower and upper bounds. You will also need to use shift registers for the values of *xlo* and *xhi*, as these are re-used between iterations of the loop.


Find xmid from xlo & xhi
Calculate f(xlo)*f(xmid)
<0?
No
Yes
Root is in upper bracket: make xlo = xmid (leave xhi alone)
Root is in lower bracket: make xhi = xmid (leave xlo alone)
e = 1/2 | xhi-xlo | < tolerance?
No
Yes
Done: root = xmid = 1/2 (xlo + xhi)