In this project you will build the hybrid solver described in class for using Newton’s method to solve the equation $f(x) = 0$. We begin with a description of the algorithm.

The input to the algorithm consists of two distinct points $a$ and $b$ that bracket a zero of $f$—i.e., at least one of $f(a)$ or $f(b)$ is zero or $f(a)f(b) < 0$. There is also a convergence criterion tol. The algorithm stops when

1. $f(a)$ or $f(b)$ is zero, or
2. $|b - a| \leq$ tol, or
3. there is no floating-point number between $a$ and $b$.

The heart of the algorithm is a while loop that exits when $a$ and $b$ satisfy one of the convergence criteria. At the beginning of the body of the loop we have two distinct points $a$ and $b$ with function values $f(a)$ and $f(b)$. These values satisfy

1. $0 < |f(a)| \leq |f(b)|$,
2. $\text{sign}(f(a)) \neq \text{sign}(f(b))$,
3. there is a floating point number between $a$ and $b$.

Note that we do not require that $a < b$.

We now compute a new iterate $c$ according to the following prescription, in which $m = (a + b)/2$.

$c$ is tentatively the Newton iterate launched from $a$ provided it is well defined and lies between $a$ (exclusive) and $m$ (inclusive). Otherwise it is tentatively $m$. Whatever the value, if $|c - a| <$ tol, $c$ is replaced by a point that lies between $a$ and $b$ and is distance $0.5 \times$ tol from $a$.

After $c$ has been determined, $f(c)$ is evaluated. If $f(c)$ is zero, an appropriate return is made. Otherwise $c$ replaces $a$ or $b$, depending on which replacement yields a new bracket. If necessary the new $a$ and $b$ are interchanged so the (2.1) is satisfied.

Figure 1 contains the specifications for a function $\text{safenewt}$ implementing this algorithm.

Code this and test this function. Be sure you try all cases; e.g., where all possibilities of the function evaluating to zero, your ability to pass additional arguments through $\text{varargin}$, etc. Twenty-four hours before the project is due, I will post a script for you to run. Return the output, along with your function and test cases, all carefully documented.
SafeNewt  Zero of a function f by bisection and Newton’s method

\[ [a, b, fa, fb] = \text{SafeNewt}(a, b, tol, func, varargin) \]
returns a bracket a, b for a zero of the function func and
the function values fa and fb at a and b. On input:

a and b are points that satisfy one of the following conditions.

1. At least one of f(a) or f(b) is zero, or
2. \( \text{sign}(f(a)) \neq \text{sign}(f(b)) \).

\( tol \) is a nonnegative convergence criterion.

\( func \) is a name or handle of the function whose zero is sought.

it has the calling sequence

\[ \text{val} = \text{func}(x, \text{job}, \text{varargin}{:}) \]

where \( x \) is the point at which to evaluate \( f \). If \( \text{job} \) is zero
\( \text{func} \) returns the value of the function, otherwise it returns
the value of its first derivative. \( \text{varargin} \) is a cell array
containing any additional arguments to \( \text{func} \).

The returned values satisfy one of the following conditions.

1. \( a == b \) and \( fa == fb = 0 \).
2. \( 0 < |b - a| \leq tol \), and \( \text{sign}(fa) \cdot \text{sign}(fb) < 0 \).
3. \( tol < |b-a| \), \( \text{sign}(fa) \cdot \text{sign}(fb) < 0 \), and there is no
   floating-point number between a and b.

Figure 1: Safe Newton method