We begin Chapter Two on Nonlinear Equations. The basic problem is to solve
\[ f(x) = 0, \quad x \in \mathbb{R}, \quad (1) \]
a nonlinear equation in one real variable. The solution \( x \) is called a root of equation (1) and a zero of the function \( f \).

Later we will briefly discuss solving
\[ \mathbf{F}(x) = 0, \quad x \in \mathbb{R}^n, \mathbf{F}(x) \in \mathbb{R}^n. \]

Note that \( x \) is my short hand for vectors in print.

To start, look at the example
\[
f(x) = \cos x - x = 0.
\]
This question is probably posed as, when does \( \cos x = x \)? Rarely, do we just submit this problem to a root finder without first “localizing” the root.

For a continuous function, the most definitive symptom of a zero is a sign change in \( f \). For instance
\[
f(0) = \cos 0 - 0 = 1, \quad f(\pi/2) = \cos \pi/2 - \pi/2 = -\pi/2.
\]
The intermediate value theorem leads us to the conclusion that \( f(x) = 0 \) for some \( x \in (0, \pi/2) \).

We write this as the following theorem

**Theorem 1** Let \( f \) be a continuous function on the interval \([c, d]\) where \( f(c)f(d) < 0 \). Then for some \( x \in (c, d) \), we have \( f(x) = 0 \).

How do we exploit this? We can get a smaller interval using bisection. Let \( c_0 = c, \ d_0 = d, \) for \( n = 0, 1, 2, \ldots \), we define a sequence of intervals \([c_n, d_n]\) all containing the zero by computing
\[
\text{mid} = (c_n + d_n)/2.
\]

There are three possible ways to choose \([c_{n+1}, d_{n+1}]\).
1. \( f(mid) = 0 \) In which case \( c_{n+1} = d_{n+1} = mid \) and we are done.

2. \( f(c_n)f(mid) < 0 \). Then

\[
d_{n+1} = mid, \quad c_{n+1} = c_n.
\]

Move the right endpoint, but not the left endpoint.

3. \( f(d_n)f(mid) < 0 \). then

\[
c_{n+1} = mid, \quad d_{n+1} = d_n.
\]

Move the left endpoint, but not the right endpoint.

Since the last two conditions are mutually exclusive, we need only check one of them. This algorithm is coded in MATLAB in the function bisect2. It is the file \texttt{www.cse.psu.edu/~barlow/cse455/bisect2.m} on the class web page. Except for the two function evaluations \((f(c_0), f(d_0))\) at the beginning, there is exactly one function evaluation per iteration.

We have that

\[
d_{n+1} - c_{n+1} = \frac{1}{2}(d_n - c_n),
\]

thus by an induction argument

\[
d_n - c_n = 2^{-n}(d_0 - c_0).
\]

A common stopping criterion is to specify that

\[
|d_n - c_n| \leq tol
\]

where \( tol \) is some tolerance. Thus we say that

\[
2^{-n}(d_0 - c_0) \leq tol.
\]

Using logarithms, that implies

\[
n \geq \log_2 (d_0 - c_0) - \log_2 tol.
\]

The machine precision in standard IEEE double is \( 2^{-53} \). A reasonable value for \( tol \) would be about \( 2^{-50} \). Thus, about 50 iterations may be necessary for convergence.
To code bisect2 in MATLAB I needed a mechanism for passing a function as an argument. MATLAB uses function handles to do this. The calling sequence for bisect2 is

\[
\gg [c,d,isroot] = \text{bisect2}(\text{phi},c0,d0,tol)
\]

where \( \text{phi} \) is a string representing the function. We put the function \( f(x) = \cos x - x \) in an m–file called cosx.m. To call it, we set the parameter using

\[
\gg \text{phi} = @\text{cosx}
\]

Inside bisect2, a function evaluation is done by the statement

\[
\gg \text{phic} = \text{feval}(\text{phi},c)
\]