

9 Finding Roots of Equations

This is a fundamental issue in numerical computation.

We will look at two methods for finding roots in this course: the bisection method and Newton's method.

These represent two different kinds of algorithms.

The **bisection method**

- is very simple to implement;
- can (almost) be guaranteed to find roots if roots exist;
- does not converge very rapidly;
- but (usually) provides an absolute measure of the error in the value of the root:

$$root_{actual} - root_{approximate} < \text{some absolute bound}$$

Newton's method

- is not hard to implement;
- can not be guaranteed to find roots even if roots exist;
- converges quite rapidly;
- but does not provide an absolute measure of the error in the root; instead, one must test whether

$$abs(f(x))$$

is very small in order to know when to terminate the iteration.

The Bisection Method

Given a function $f(x)$ of which we wish to find roots in an interval $a \leq \text{root} \leq b$, then if $f(a)$ and $f(b)$ are of opposite signs, and the function is continuous, there must be a root somewhere in the interval from a to b .

```
if( f(x_lower) * f(x_upper) < 0.0 )
{
    x_middle = (x_lower + x_upper)/2.0
    if( f(x_lower) * f(x_middle) < 0.0 )
    {
        x_upper = x_middle // use x_lower and x_middle as new endpoints
    }
    else
    {
        x_lower = x_middle // use x_middle and x_upper as new endpoints
    }
}
```

- We repeat this until $\text{abs}(x_{\text{upper}} - x_{\text{lower}}) < \text{epsilon}$
- We stop on an *absolute* condition on the root itself.
- We choose the midpoint of the last interval used $(x_{\text{upper}} + x_{\text{lower}}) / 2.0$ as the approximation to the root.
- Obviously this approximate root cannot be any further from the actual root than one-half the length of the interval of which it is the midpoint, which is $(x_{\text{upper}} + x_{\text{lower}}) / 4.0$
- Question: What if we don't have x -values for which $f(x)$ is of opposite signs?
- Then we will have to search for values for which we do get opposite signs.