Introduction to Scientific Computing
Lecture 4

Professor Hanno Rein

Last updated: October 2, 2017

Root finding algorithms

0.1 Intermediate Value Theorem

The intermediate value theorem is a fundamental mathematical theorem which we will not prove, but it will be important for the methods we derive and you will need to understand its consequences.

Let $I = [a, b]$ be an interval with real numbers a, b and let f be a continuous function from the interval I into the space of real numbers, $f : I \to \mathbb{R}$. For any number c for which

$$f(a) < c < f(b),$$

then there is a number $d \in [a, b]$ such that

$$f(d) = c.$$

0.2 The problem of root finding

We have already encounter one problem of root finding, the least square fit. In that case, we found the root of the equation

$$\frac{\partial S(a)}{\partial a} = 0$$

Because we only considered a *linear* least square fit, we were able to solve the problem exactly using Gaussian elimination. In general it will not be possible to solve a root finding problem this easily (or at all!).

So what do we mean by root finding? A root finding algorithm finds the value x for which a given function $f(x)$ is zero. In the least square fit example, the function $f(x)$ was $\partial S/\partial a$ and the variable x was the vector a.

The following might seem obvious but its worth pointing out. The way we defined root finding, we are looking for a function argument that makes a function evaluate to zero. However, this is equivalent to trying to solve the equation

$$f(x) = g(x)$$

for two arbitrary function f and g. We just need to bring one to the other side to get it back into the canonical form $f(x) - g(x) = 0$.

0.3 Bisection method

All the methods we will discuss to solve the root finding problem are iterative. That means we start with a guess and improve upon our guess during every iteration. The bisection method is one such example.

Let's formulate the problem a bit more precisely. You are given a one dimensional real function on the interval $I = [a, b]$. You can assume that the function is continuous. The problem is then to find the value $d \in [a, b]$ for which $f(d) = c$. You are given c, usually we have $c = 0$.

The bisection method works as follows. We first divide the interval $[a, b]$ into two intervals $[a, \frac{a+b}{2}]$ and $[\frac{a+b}{2}, b]$. We then evaluate the function at the middle point $\frac{a+b}{2}$. If the value $f(\frac{a+b}{2})$ is smaller than c, then we know that our answer must lie in the upper interval $[\frac{a+b}{2}, b]$. Similarly, if the value $f(\frac{a+b}{2})$ is larger than c, then we know that our answer must lie in the lower interval $[a, \frac{a+b}{2}]$. We now have a better estimate of the value d. By simply repeating the process, we can make it more and more accurate. Because we divide the interval in half every time, the method converges quickly.

Below is an illustration of the bisection method and an implementation in pseudo code.

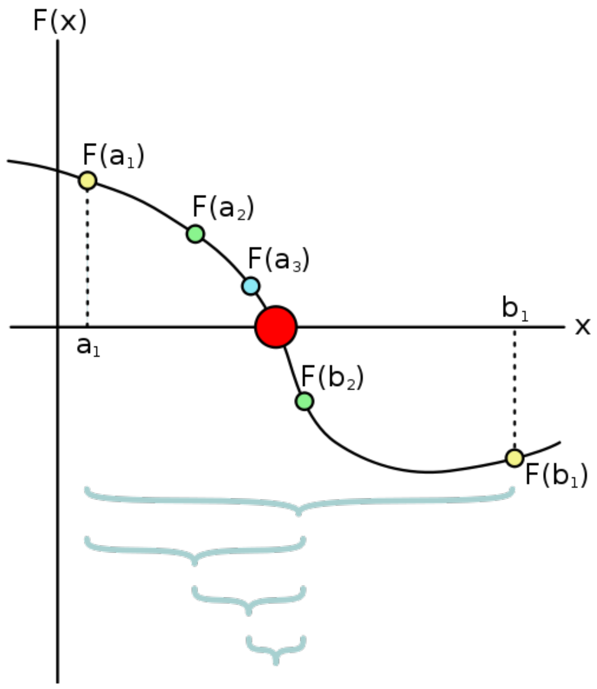

Figure 1: Bisection methods. Source: Wikipedia.

```
set a
set b
fa = f(a)
fb = f(b)
while ( (b-a) > epsilon ){
    m = (a+b)/2
    fm = f(m)
    if ( (fm>0 and fa<0) or (fm<0 and fa>0) ){
        b = m
        fb = f(b)
    }else{
        a = m
        fa = f(a)
    }
}
print [a:b]
```

Code 1: Pseudo code of the bisection method.

Ok. So the method improves the result at every iteration and the interval gets small. The question is: Using standard double floating point precision, how many iteration steps do we roughly need to converge to machine precision? The answer is 52 as there are 52 bits in the mantissa.

Note that we didn't use the actual value of $f(\frac{a+b}{2})$, just its sign. We're throwing away information that we could have used. There are much better methods than the bisection method that make use of this information.