MATH 320: CLASS NOTES

1. Roots - Chapter 5

Let's solve the equation $f(x) = 0$.

1.1. **Bracketing Methods.**

Definition 1.1. Bracketing methods are any root-finding algorithms that start with an interval containing a root.

Algorithm 1.2 (Incremental search)**.** *INPUT: function f, search interval $[l, r]$.*

(1) Divide the interval into N sub-intervals, with points $l = x_0 < \ldots < x_N = r$ evenly spaced in the interval.

(2) Compute $f(x_i)$ for all i.

(3) List all i for which $\operatorname{sign}(f(x_i)) \neq \operatorname{sign}(f(x_{i+1}))$.

OUTPUT: List of brackets $\{(x_i, x_{i+1})\}$ flagged in step 3.

Question 1.3 (Accuracy)**.** Does this algorithm catch all the roots? What roots will this algorithm miss? Will we ever catch extra roots?

It misses roots when N is not large enough.
It does not catch roots whose neighborhood all has one sign.
If a bracket is flagged, there will be a root in it. Why?

Theorem 1.4 (Intermediate Value Theorem)**.** *If f is a continuous function on $[a, b] \subset \mathbb{R}$, $f(a) < 0$, and $f(b) > 0$; then there exists a point $c \in [a, b]$ such that $f(c) = 0$.*

Question 1.5 (Complexity)**.** How long does the algorithm take?

It evaluates the function $N + 1$ times, and checks the sign on each value. The complexity is $O(N)$.

Question 1.6 (Error)**.** How close to the true roots will the roots found by our algorithm be?

Take the approximation of a given root to be in the middle of the flagged bracket. Our error is then bounded by $(l - r)/2N$.

The bisection method is an iterative form of the incremental search.

Algorithm 1.7 (Bisection Method)**.** *INPUT: function f, search interval $[l, r]$ such that $\operatorname{sign}(f(l)) \neq \operatorname{sign}(f(r))$, desired precision ϵ.*

(1) If $l - r < 2\epsilon$, stop.

(2) Divide the interval into 2 equal sub-intervals: $l < m < r$.

(3) Compute $f(m)$.

(4) If $\text{sign}(f(m)) = \text{sign}(f(l))$, repeat with interval $[m, r]$; else, repeat with interval $[l, m]$.

OUTPUT: Final bracket endpoints.

Accuracy: Catches exactly one root.

Complexity: Will stop when $l - r < 2\epsilon$ since $l - r$ shrinks by half every time, the length of the interval at iteration N is $(1/2)^N (l - r)$.

$$(1/2)^N (l - r) < 2\epsilon \iff N > \log_2((l - r)/2\epsilon)$$

Error: By design the error is at most ϵ.

Question 1.8 (Accuracy)**.** Does this algorithm catch all the roots? What roots will this algorithm miss? Will we ever catch extra roots?

It misses roots when N is not large enough.
It does not catch roots whose neighborhood all has one sign.
If a bracket is flagged, there will be a root in it. Why?

Theorem 1.9 (Intermediate Value Theorem)**.** *If f is a continuous function on $[a, b] \subset \mathbb{R}$, $f(a) < 0$, and $f(b) > 0$; then there exists a point $c \in [a, b]$ such that $f(c) = 0$.*

Question 1.10 (Complexity)**.** How long does the algorithm take?

It evaluates the function $N + 1$ times, and checks the sign on each value. The complexity is $O(N)$.

Question 1.11 (Error)**.** How close to the true roots will the roots found by our algorithm be?

Take the approximation of a given root to be in the middle of the flagged bracket. Our error is then bounded by $(l - r)/2N$.