

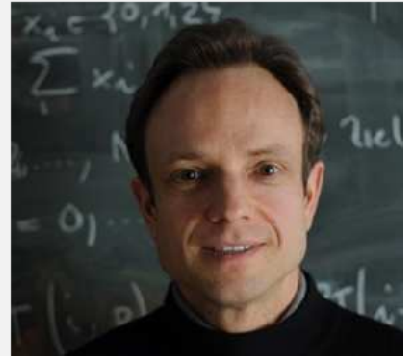
Course Video

Section 8.4.1: Finding Zeros of Scalar Functions: Bisection

Prof. R. Hiptmair, SAM, ETH Zurich

Date: December 1, 2020

(C) Seminar für Angewandte Mathematik, ETH Zürich



Dependencies. Section 8.2.1, Section 8.2.2

Duration: 10 minutes

VIII Iterative Methods for Non-Linear Systems of Equations

8.4. Finding Zeros of Scalar Functions

Now : $n = 1$

$$F : I \subset \mathbb{R} \longrightarrow \mathbb{R}$$

$$\text{Find : } x^* \in \mathbb{R} : F(x^*) = 0$$

②

8.4.1 Bisection

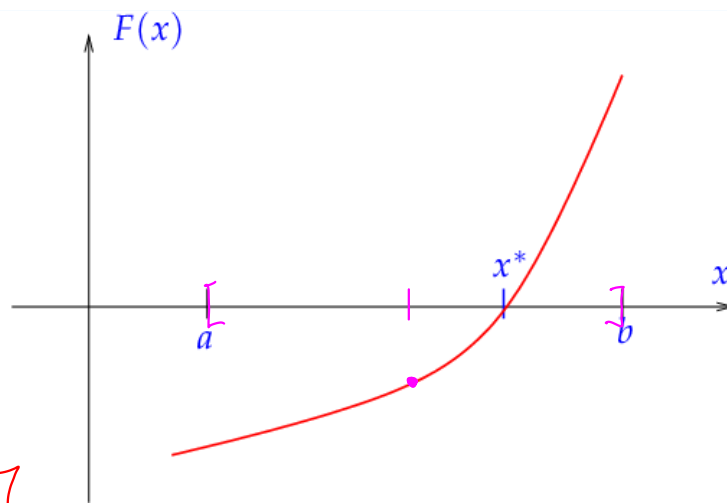
Assume: F continuous

Input: $a, b \in I$ such that

$$F(a)F(b) < 0. \quad (\text{different signs!})$$

► $\exists x^* \in]\min\{a, b\}, \max\{a, b\}[$:
 $F(x^*) = 0,$

[Intermediate value theorem]



Idea: Enclose x^* in geometrically decreasing intervals $[a^{(k)}, b^{(k)}]$ on which f changes sign: $f(a^{(k)})f(b^{(k)}) \leq 0$

$$x^{(k)} := a^{(k)}$$

C++ code 8.4.1.2: Bisection method for solving $F(x) = 0$ on $[a, b]$

```

2 // Searching zero of F in [a,b] by bisection
3 template <typename Func, typename Scalar>
4 Scalar bisection(Func&& F, Scalar a, Scalar b, Scalar tol)
5 {
6     if (a > b) std::swap(a,b); // sort interval bounds
7     if (F(a)*F(b) > 0) throw "f(a) and f(b) have same sign";
8     static_assert(std::is_floating_point<Scalar>::value,
9                   "Scalar must be a floating point type");
10    int v = F(a) < 0 ? 1 : -1;
11    Scalar x = (a+b)/2; // determine midpoint
12    // termination, relies on machine arithmetic if tol = 0
13    while (b-a > tol) { //
14        assert(a <= x && x <= b); // assert invariant
15        // sgn(f(x)) = sgn(f(b)), then use x as next right boundary
16        if (v * F(x) > 0) b = x;
17        // sgn(f(x)) = sgn(f(a)), then use x as next left boundary
18        else a = x;
19        x = (a+b)/2; // determine next midpoint
20    }
21    return x;
22 }
```

"Linear-type" convergence, rate = $\frac{1}{2}$

$$|b^{(k)} - a^{(k)}| \leq 2^{-k} |b - a|$$

$$\triangleright |x^{(k)} - x^*| \leq 2^{-k} |b - a|$$

- + Guaranteed convergence: **robust method**
- + F-evaluations only: "derivative-free"
- + Works $\forall$ continuous F

③

▷ Review questions 8.4.1.4

4

5

7