Root Finding: Bisection Method

ME 2004

Outline

- 1.1: Bisection Method

Bisection Method

- Bisection Method: application of the Intermediate Value Theorem
- Premise: the root lies somewhere in a user-specified initial interval. The interval is narrowed down until it “hugs” the root.
- Called a *bracketing method* because the user must supply the algorithm with a bracket which contains the root
  - Contrast: Newton-Raphson method is an *open method* because it only requires an initial estimate (not a bracket) of the root
- Assumption: $f(x)$ is real and continuous

Bisection Method

- Whenever we mention a function $f(x)$ in root finding, we imply a function in "$f(x) = 0$" form
  - Roots vs. intercepts – there's a difference! (Root Finding Intro video)

- *Example*: If we want to compute the roots of $x^3 = 27$:

$$f(x) = x^3 - 27$$

$$\cancel{f(x) = x^3}$$

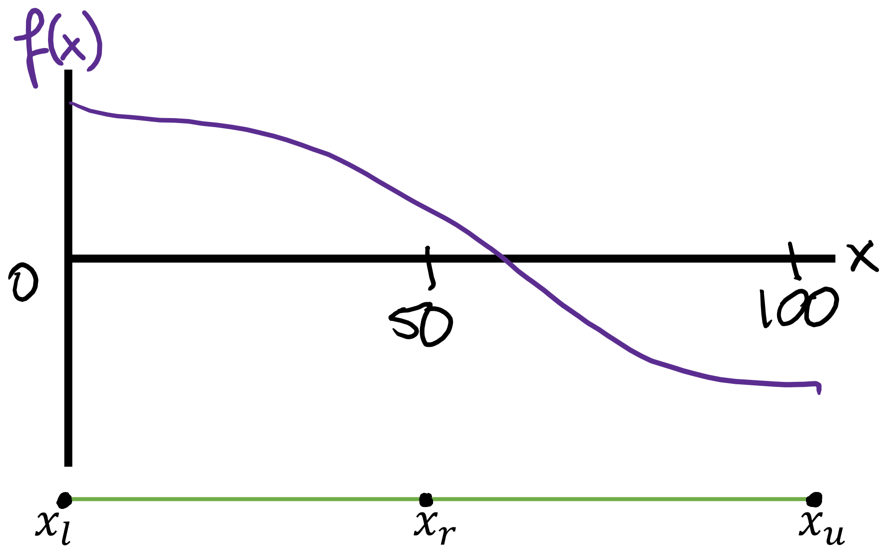
Bisection Method

“such that”

- Brief algorithm overview:
  1) Specify an initial interval: $[x_l \;\; x_u] \ni sign\big(f(x_l)\big) \neq sign\big(f(x_u)\big)$
  2) Assume the root x_r lies at the midpoint of the interval
  3) Compute error*. If error is too large…
  4) Divide the initial interval into two (equal) subintervals: $[x_l \;\; x_r]$ and $[x_r \;\; x_u]$
  5) Evaluate $f(x_l)$ and $f(x_r)$ and determine which of the two subintervals contains a sign change
  6) Assign new x_l, x_u accordingly and iterate until the stopping criterion e_s is met

*error can be a tolerance in x: $err = 100\% * \left|\frac{x_{r,current} - x_{r,previous}}{x_{r,current}}\right| \leq e_s$

or a tolerance in y: $err = \left|f\left(x_{r,current}\right)\right| \leq e_s$



1st iteration:

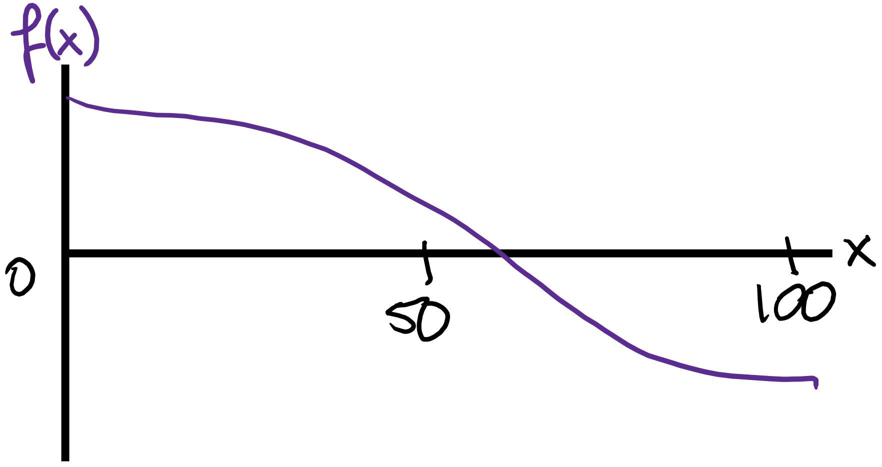
$$x_l = 0, x_u = 100$$
$$x_r = \frac{100+0}{2} = 50$$

Subintervals: [0 50] and [50 100]

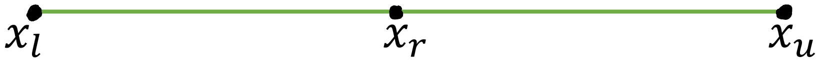
$f(x_l) > 0, f(x_r) > 0$ → no sign change in [0 50]

→ Discard [0 50] (no sign change) and keep [50 100] (sign change).

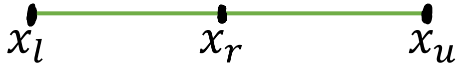
New $x_l = 50$, new $x_u = 100$.



1st iteration:



2nd iteration:



$$x_l = 50, x_u = 100$$

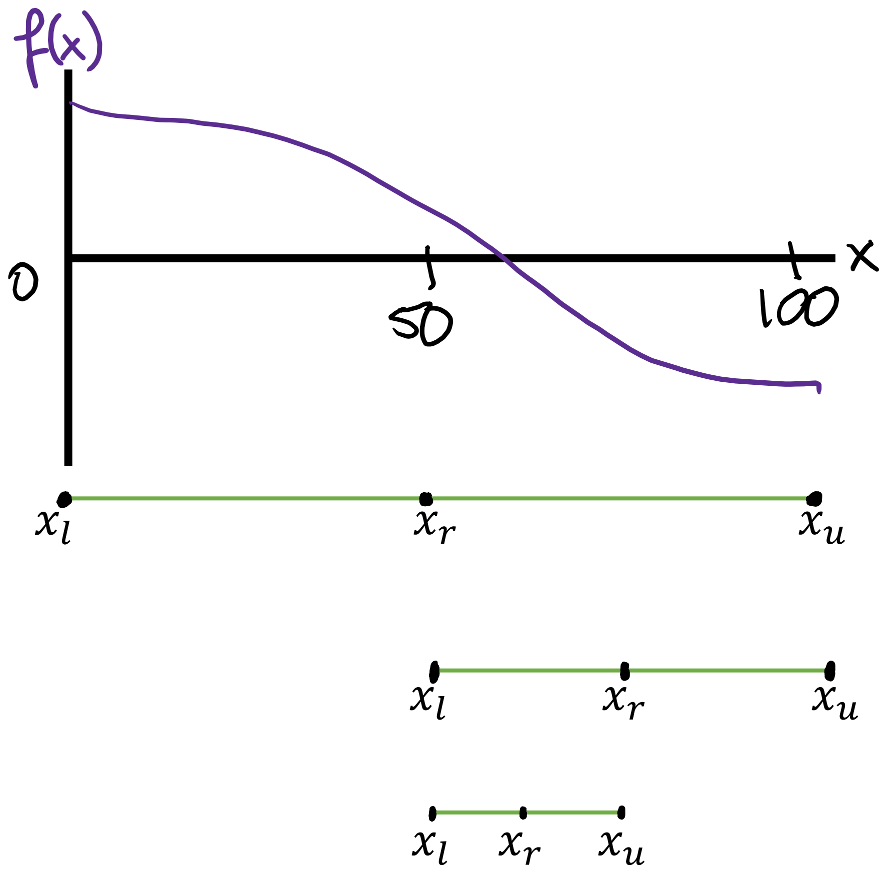
$$x_r = \frac{100 + 50}{2} = 75$$

Subintervals: [50 75] and [75 100]

$f(x_l) > 0, f(x_r) < 0 \rightarrow$ sign change in [50 75]

→ Discard [75 100] (no sign change) and keep [50 75] (sign change).

New $x_l = 50$, new $x_u = 75$.



1st iteration:

2nd iteration:

3rd iteration:

$$x_l = 50, x_u = 75$$

$$x_r = \frac{50 + 75}{2} = 62.5$$

Subintervals: [50 62.5] and [62.5 75]

$f(x_l) > 0, f(x_r) < 0$ → sign change in [50 62.5]

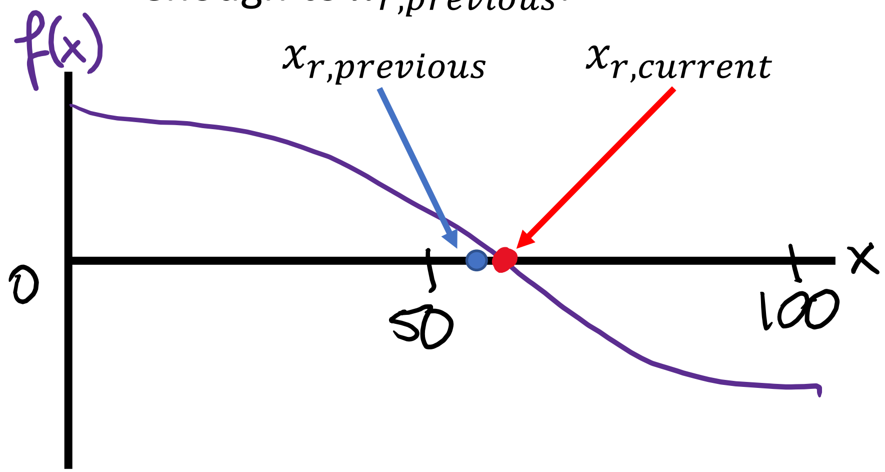
→ Discard [62.5 75] (no sign change) and keep [50 62.5] (sign change).

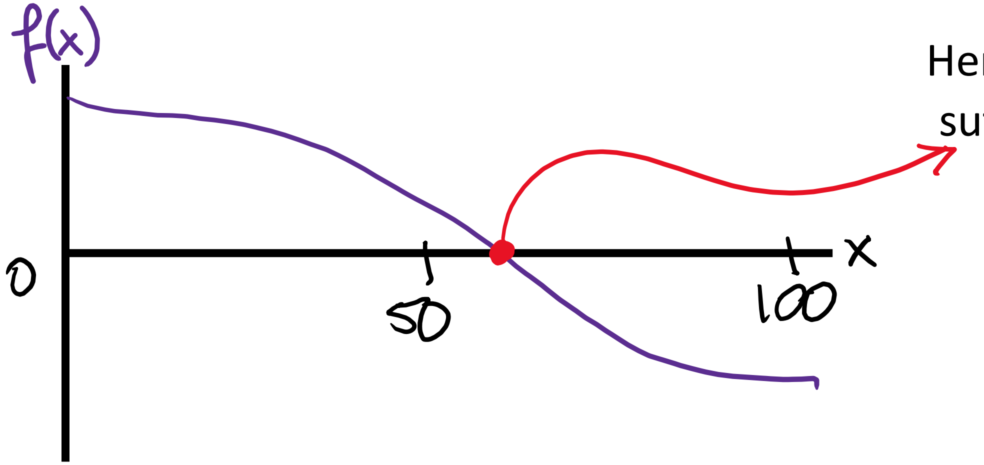
New $x_l = 50$, new $x_u = 62.5$.

Bisection Method

- Stopping criterion (e_s): condition which terminates the search if met
  - Relative error (tolerance in x): $err = 100\% * \left|\frac{x_{r,current} - x_{r,previous}}{x_{r,current}}\right| \leq e_s$
  - Absolute error (tolerance in y): $err = \left|f\left(x_{r,current}\right)\right| \leq e_s$

Here, $x_{r,current}$ is close enough to $x_{r,previous}$.





Here, $\left|f\left(x_{r,current}\right)\right|$ is sufficiently close to 0.

Bisection Method

- Number of iterations required to attain an absolute error $E_t = |true - approx|$ can be computed a priori
- After 1 iteration, the original interval $\Delta x = [x_l\ x_u]$ becomes $\frac{\Delta x}{2}$
- After 2 iterations, the original interval is reduced to $\frac{\Delta x}{4}$ ...

$$E_{t,n} = \frac{|x_u - x_l|}{2^n} \rightarrow \boxed{n \leq \log_2\left(\frac{|x_u - x_l|}{E_t}\right)}$$

- Error behaves linearly (linear convergence): $e_{current} = \frac{e_{previous}}{2}$

Bisection Method

Pros

- Guaranteed to converge (if the initial guesses are valid)
- Logical, easy-to-follow algorithm
- Can compute the max number of iterations needed ahead of time

Cons

- Slow (linear convergence)
- Computationally expensive

Summary

- Bisection Method requires 2 initial guesses such that $sign\left(f(x_l)\right) \neq sign\left(f(x_u)\right)$
- Bisection is guaranteed to converge (if initial guesses are valid), albeit slowly
- Max number of iterations required to attain prespecified true error can be computed ahead of time