Computational Engineering - Numerical Methods

Dr. Kalidas Yeturu

Roots of functions

- Motivation
  - Very generic, applicable in different engineering contexts
  - Solving gradient function equals 0
  - Plotting of contours
- Some of the popular examples
  - Matlab – roots() and fzero() functions
  - Python – scipy.optimize.root()

Definition and examples

- Given $f(x)$
- Roots are a set of values of x, $\exists x : \{x : f(x) = 0\}$

- Simple equations
  - $f(x) = mx + c$
  - $f(x) = ax^2 + bx + c \rightarrow$

$$x = \left\{\frac{-b - \sqrt{b^2 - 4ac}}{2a}, \frac{-b + \sqrt{b^2 - 4ac}}{2a}\right\}$$

Need *sophisticated methods* for finding roots of higher order and transcendental functions

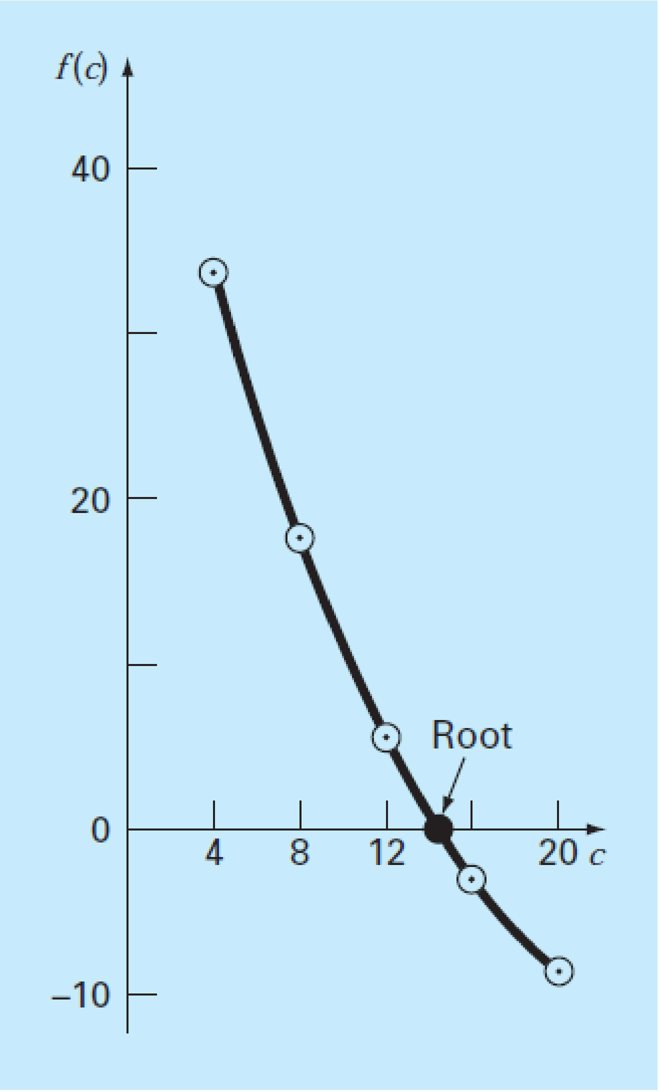
- Higher order polynomials
  - $f(x) = 5\mathrm{x}^2 - \mathrm{x}^3 + 7\mathrm{x}^6$

- Transcendental equations
  - $f(x) = \ln(\mathrm{x}^2) - 1$
  - $f(x) = \mathrm{e}^{-0.2x}\sin(3\mathrm{x} - 0.5)$

- Types of methods
  - Single real root detection
  - Determining all roots both real and complex

Root detection by visual inspection of plots

- $f(c) = \frac{gm}{c}\left(1 - e^{\frac{-c}{m}*t}\right) - v$
- $t = 10, g = 9.8, v = 40, m = 68.1$
- Plot and examine where it intersects X axis
- Root ~ 14.75



Issues

- Error prone
- Subjective
- Not scalable (imagine, we need it as a routine invoked 1 million times)
- Difficult for multivariate scenarios

c	f(c)
4	34.115
8	17.653
12	6.067
16	−2.269
20	−8.401

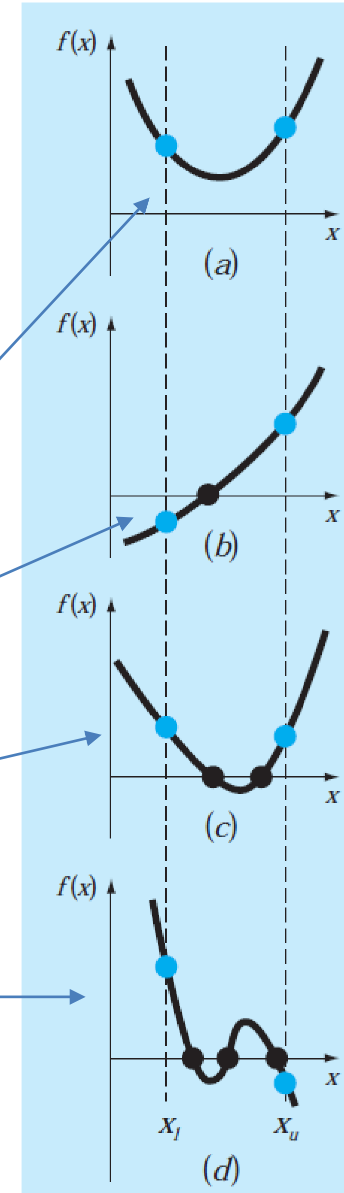
Algorithms for root finding

- Bracketing methods
  - Where root exists in an *interval*
  - Bisection
  - Regula Falsi
- Open methods
  - Where *starting* from a *single point*, a sequence of steps will lead to root
  - Simple Fixed Point
  - Newton-Raphson
  - Secant Method
- Iterative in nature
  - New value at time step *t+1* depends of value at time step *t*
- Special cases and convergence aspects

Root finding – Bracketing methods

- Given an interval of coordinates on X axis
- Determine a point in the closed interval as root

- Case (a) – Root does not exist in the interval
- Case (b) – Single root exists
- Case (c) – Two roots exist
- Case (d) – Multiple roots exist



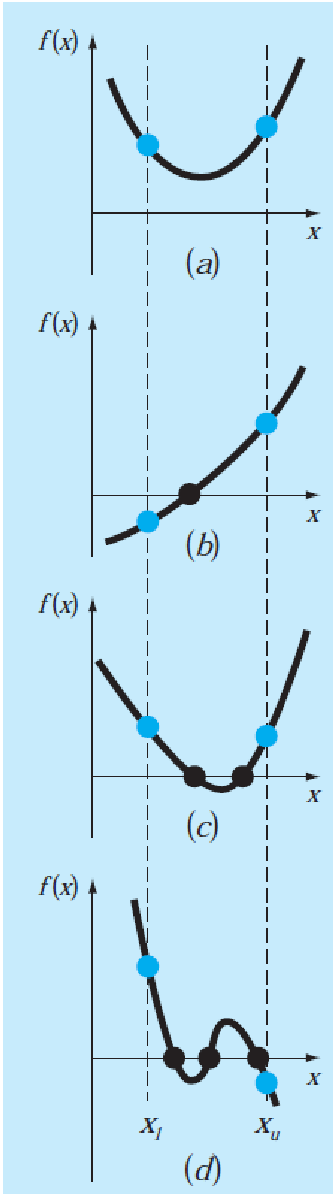
Bracketing methods…

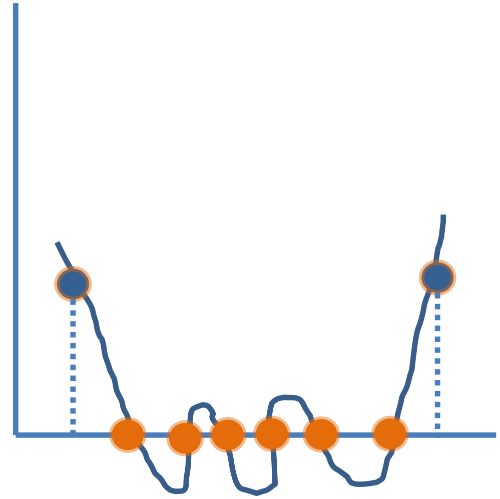
Same sign
0 roots

Opposite sign
1 roots

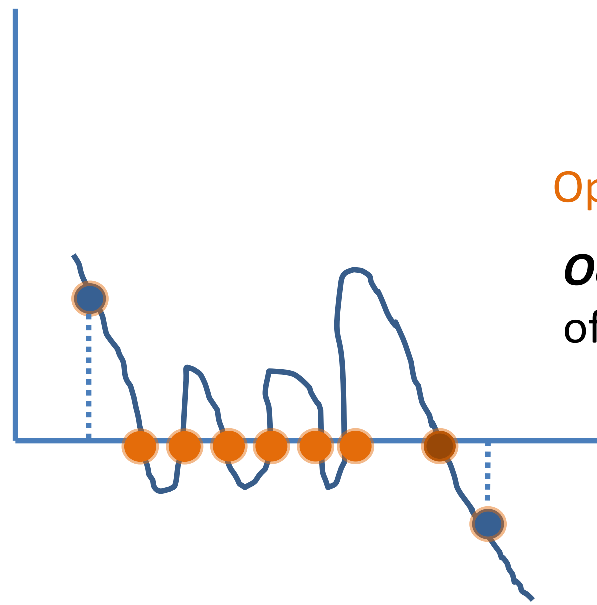
Same sign
2 roots

Opposite sign
3 roots





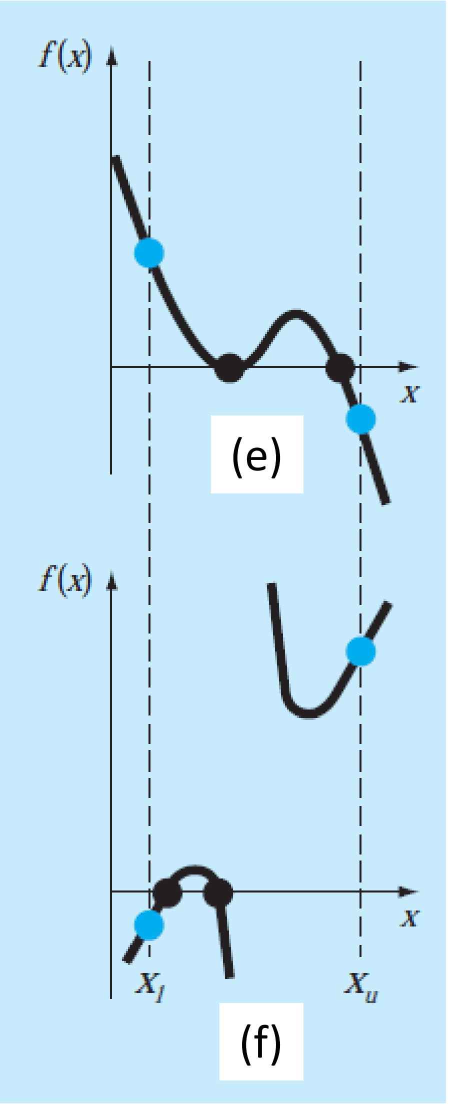
Same sign
Even number of roots

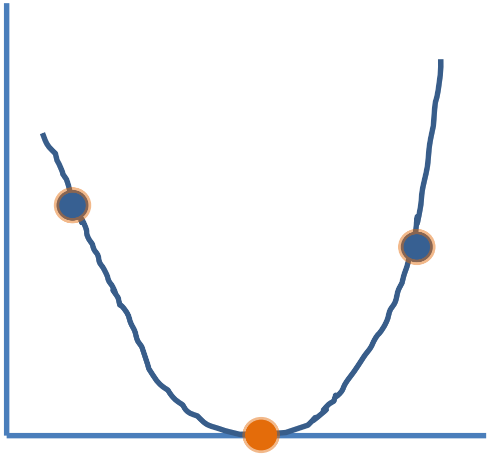


Opposite sign
Odd number of roots

Bracketing methods…

- Opposite signs
  - Case (e): Function is tangential to X axis :Two roots
  - Case (f): Function is discontinuous: Two roots
- Same sign, function touches X axis
  - Case (g) : One root
- These are special cases, need to handled separately



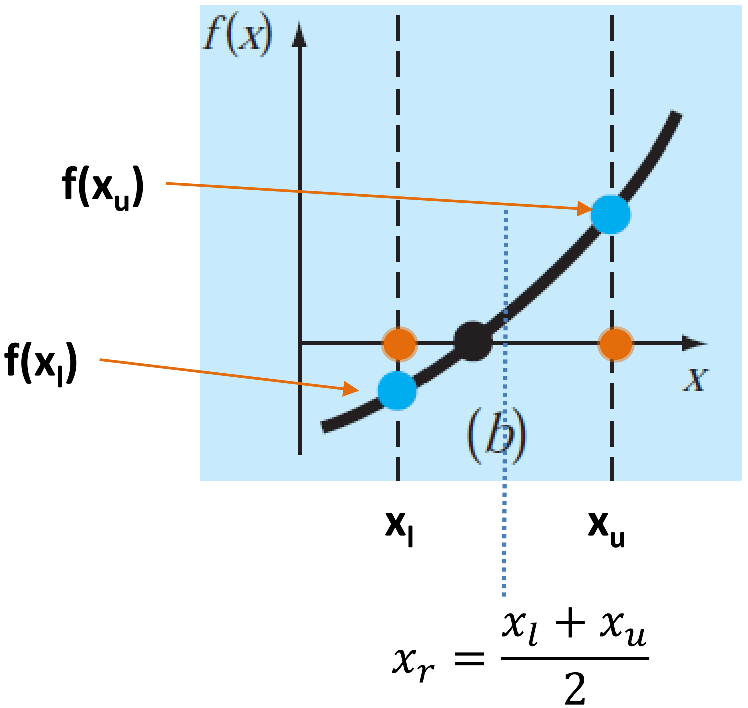


(g)

Bracketing methods – Bisection/Bolzano method

1. Choose xl and xu such that function changes sign $f(x_l) * f(x_u) < 0$
2. Determine mid point $x_r = \frac{x_l + x_u}{2}$
3. Evaluate cases
   - If $f(x_l) * f(x_r) < 0$
     Root lies between [x_l and x_r]
     Set xu = xr
     Goto *Step 2*
   - If $f(x_l) * f(x_r) > 0$
     Root lies between [x_r and x_u]
     Set xl = xr
     Goto *Step 2*
   - If $f(x_l) * f(x_r) = 0$
     x_r is the root
     Break from loop

- Report x_r



Bisection method…

Input initial xr

```
FUNCTION Bisect(xl, xu, es, imax, xr, iter, ea)
  iter = 0
  DO
    xrold = xr
    xr = (xl + xu) / 2
    iter = iter + 1
    IF xr ≠ 0 THEN
      ea = ABS((xr − xrold) / xr) * 100
    END IF
    test = f(xl) * f(xr)
    IF test < 0 THEN
      xu = xr
    ELSE IF test > 0 THEN
      xl = xr
    ELSE
      ea = 0
    END IF
    IF ea < es OR iter ≥ imax EXIT
  END DO
  Bisect = xr
END Bisect
```

Avoid divide by 0

Set error to 0

Error is small
Number of iterations is over

Iteration	x_l	x_u	x_r	ε_a (%)	ε_t (%)
1	12	16	14		5.279
2	14	16	15	6.667	1.487
3	14	15	14.5	3.448	1.896
4	14.5	15	14.75	1.695	0.204
5	14.75	15	14.875	0.840	0.641
6	14.75	14.875	14.8125	0.422	0.219

$$\varepsilon_a = \left|\frac{x_r^{new} - x_r^{old}}{x_r^{new}}\right| * 100\%$$

$$\varepsilon_t = \left|\frac{true - x_r^{new}}{true}\right| * 100\%$$

Bisection method…!

2n evaluations of f(x)

```
FUNCTION Bisect(xl, xu, es, imax, xr, iter, ea)
  iter = 0
  DO
    xrold = xr
    xr = (xl + xu) / 2
    iter = iter + 1
    IF xr ≠ 0 THEN
      ea = ABS((xr − xrold) / xr) * 100
    END IF
    test = f(xl) * f(xr)
    IF test < 0 THEN
      xu = xr
    ELSE IF test > 0 THEN
      xl = xr
    ELSE
      ea = 0
    END IF
    IF ea < es OR iter ≥ imax EXIT
  END DO
  Bisect = xr
END Bisect
```

Two times f(x) evaluations

Improved Bisection method

n+1 evaluations of f(x)

```
FUNCTION Bisect(xl, xu, es, imax, xr, iter, ea)
  iter = 0
  fl = f(xl)
  DO
    xrold = xr
    xr = (xl + xu) / 2
    fr = f(xr)
    iter = iter + 1
    IF xr ≠ 0 THEN
      ea = ABS((xr − xrold) / xr) * 100
    END IF
    test = fl * fr
    IF test < 0 THEN
      xu = xr
    ELSE IF test > 0 THEN
      xl = xr
      fl = fr
    ELSE
      ea = 0
    END IF
    IF ea < es OR iter ≥ imax EXIT
  END DO
  Bisect = xr
END Bisect
```

One time evaluation ***before*** the loop

Single invocation of f(x) ***inside*** the loop

Remember the value

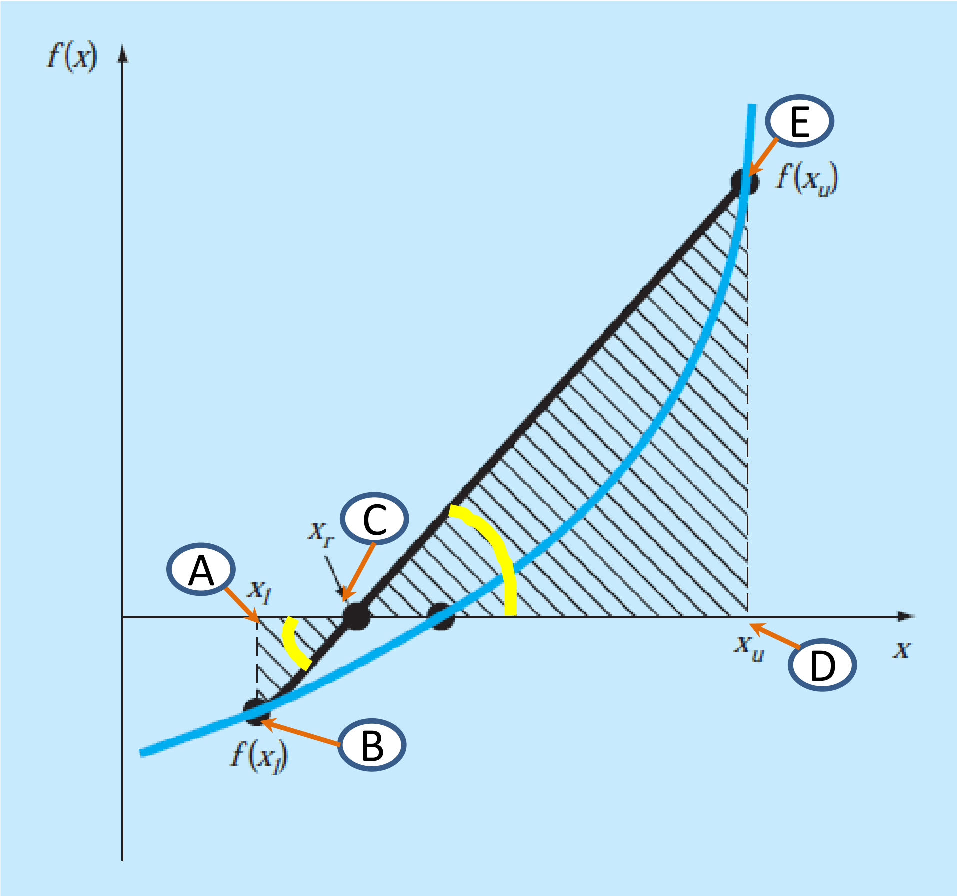
Bracketing methods – Regula Falsi Method

- Consider triangles ΔABC and ΔCDE
- Vertical angles are same $\langle ACB = \langle ECD$
- Ratio of opposite side by adjacent sides are same → $\frac{ED}{CD} = \frac{AB}{AC}$
- Translating the above ratio to function values and X coordinates

$$\frac{f(x_u)}{(x_u - x_r)} = \frac{-f(x_l)}{(x_r - x_l)}$$

- After rearranging terms, we have

$$x_r = x_u - \frac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)}$$



Idea – As the bracket is close to the root, ***curve segment is nearly a straight line…***

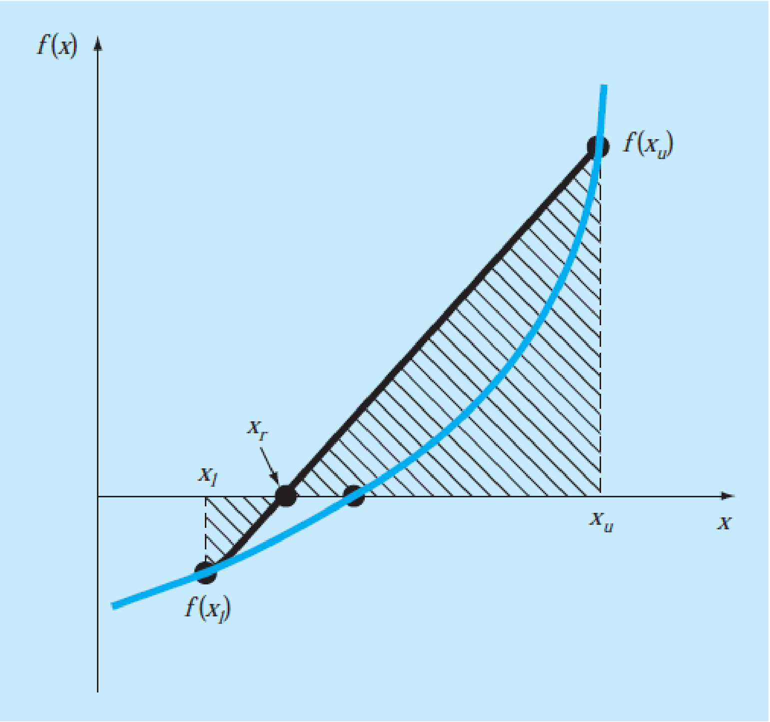
Regula Falsi method…

1. Choose xl and xu such that function changes sign $f(x_l) * f(x_u) < 0$
2. Determine mid point

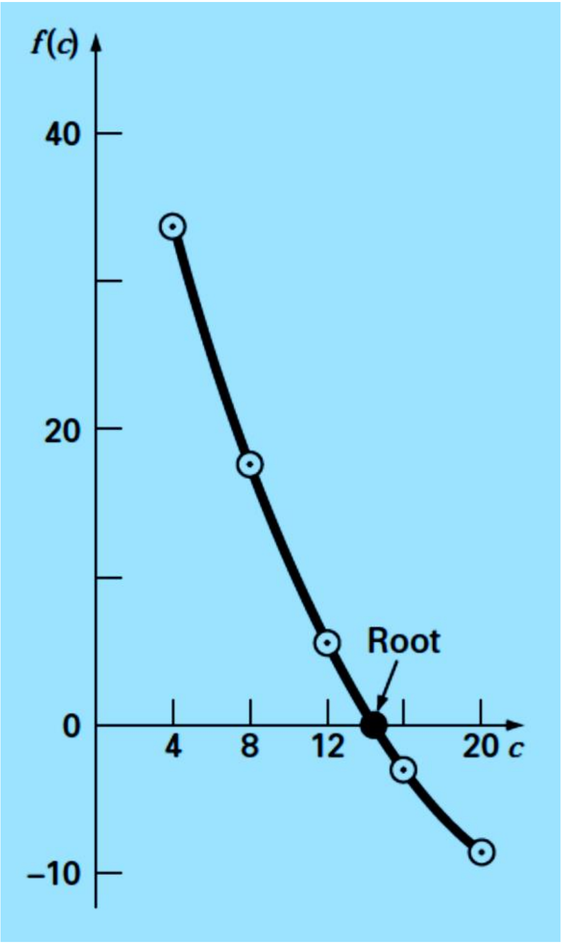
$$x_r = x_u - \frac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)}$$

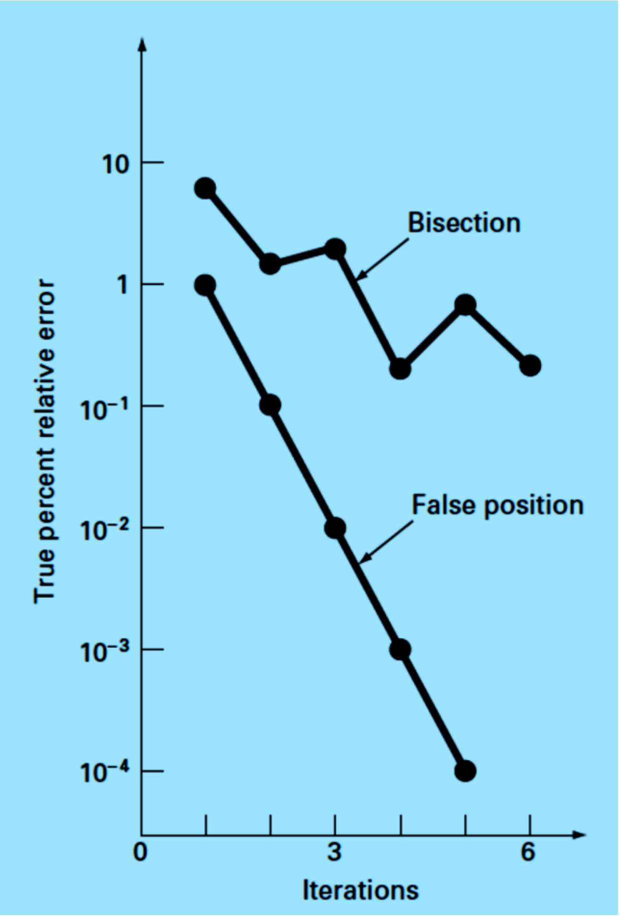
1. Evaluate cases
   - If $f(x_l) * f(x_r) < 0$
     Root lies between [x_l and x_r]
     Set xu = xr
     Goto *Step 2*
   - If $f(x_l) * f(x_r) > 0$
     Root lies between [x_r and x_u]
     Set xl = xr
     Goto *Step 2*
   - If $f(x_l) * f(x_r) = 0$
     x_r is the root
     Break from loop

- Report x_r

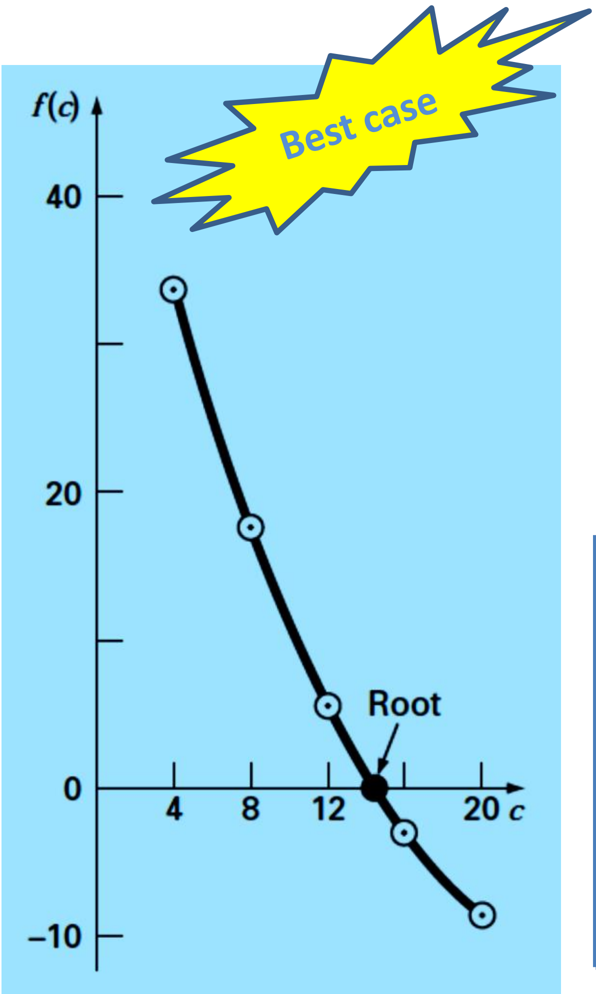


Regula Falsi method…

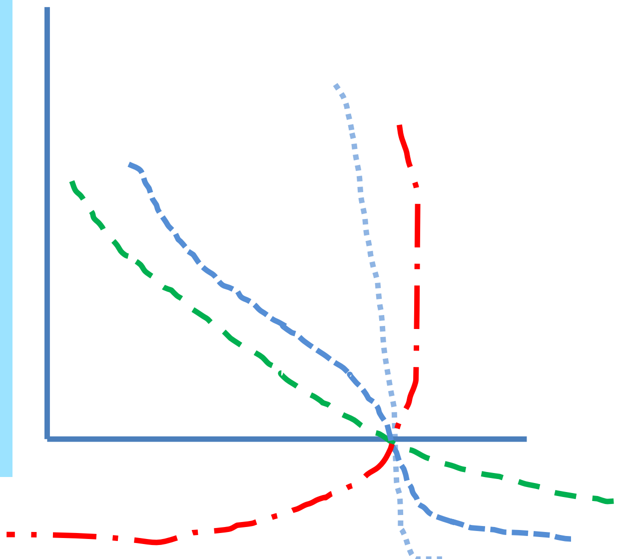


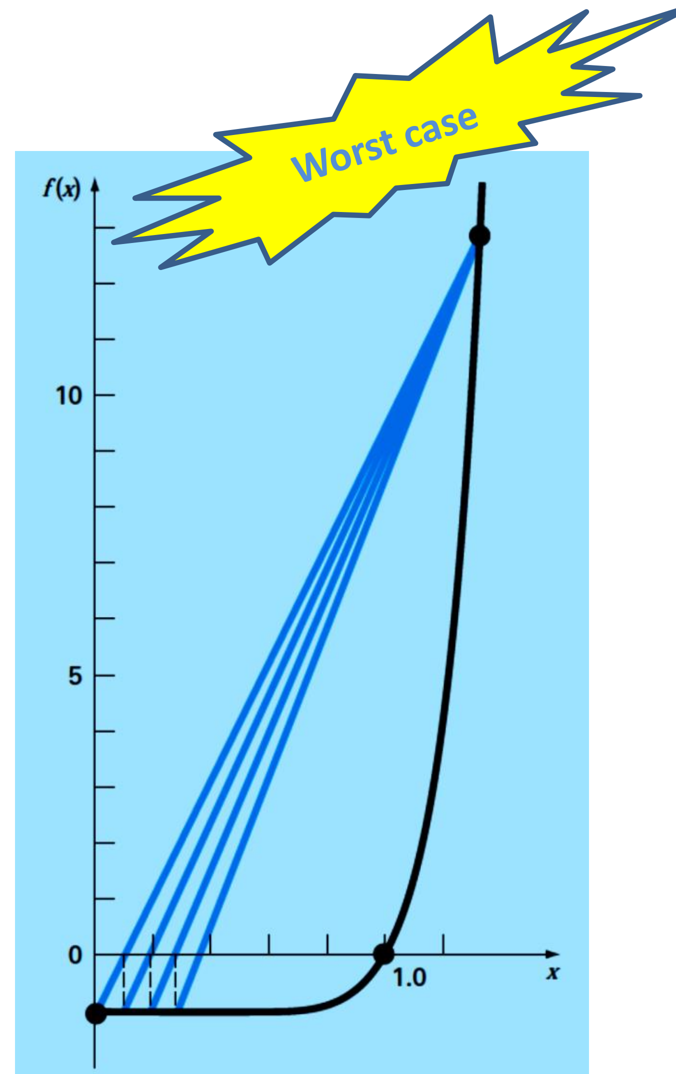


Regula Falsi method…



Different curves
have different convergence





Regula Falsi method…

Naïve approach

1. Choose xl and xu such that function changes sign $f(x_l) * f(x_u) < 0$
2. Determine mid point

$$x_r = x_u - \frac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)}$$

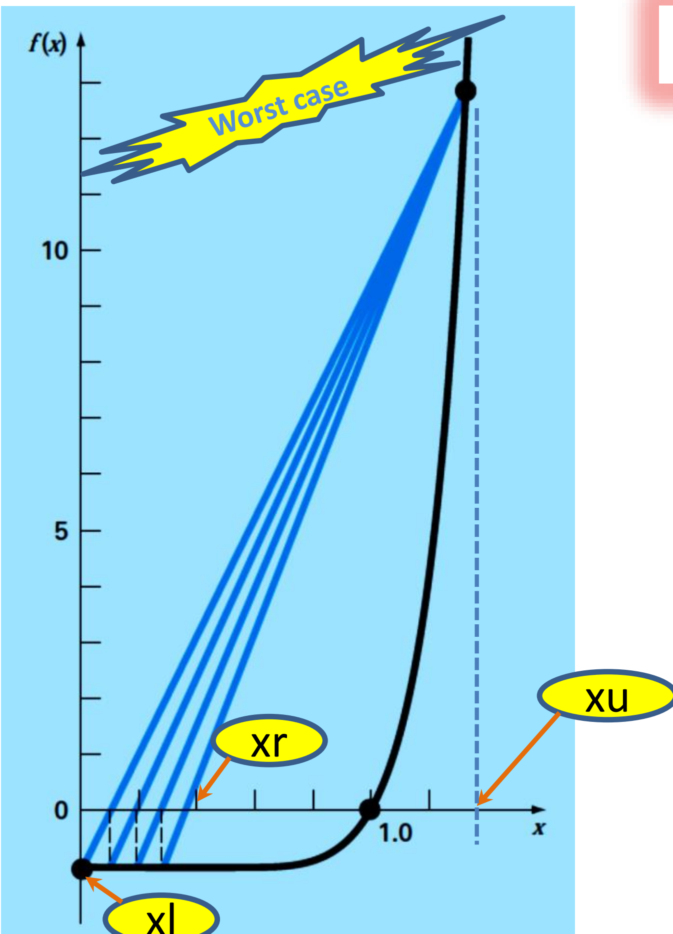
1. Evaluate cases
   - If $f(x_l) * f(x_r) < 0$
     Root lies between [x_l and x_r]
     Set xu = xr
     Goto *Step 2*
   - If $f(x_l) * f(x_r) > 0$
     Root lies between [x_r and x_u]
     Set xl = xr
     Goto *Step 2*
   - If $f(x_l) * f(x_r) = 0$
     x_r is the root
     Break from loop

- Report x_r

Enhanced approach

```
FUNCTION ModFalsePos(xl, xu, es, imax, xr, iter, ea)
  iter = 0
  fl = f(xl)
  fu = f(xu)
  DO
    xrold = xr
    xr = xu − fu * (xl − xu) / (fl − fu)
    fr = f(xr)
    iter = iter + 1
    IF xr <> 0 THEN
      ea = Abs((xr − xrold) / xr) * 100
    END IF
    test = fl * fr
    IF test < 0 THEN
      xu = xr
      fu = f(xu)
      iu = 0
      il = il + 1
      If il ≥ 2 THEN fl = fl / 2
    ELSE IF test > 0 THEN
      xl = xr
      fl = f(xl)
      il = 0
      iu = iu + 1
      IF iu ≥ 2 THEN fu = fu / 2
    ELSE
      ea = 0
    END IF
    IF ea < es OR iter ≥ imax THEN EXIT
  END DO
  ModFalsePos = xr
END ModFalsePos
```

Regula Falsi method...



x_u is not moving

...
...

```
ELSE IF test > 0 THEN
  xl = xr
  fl = f(xl)
  il = 0
  iu = iu + 1
  IF iu ≥ 2 THEN fu = fu / 2
```

...
...

$$f(x_l) < 0$$

$$x_r = x_u - \frac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)}$$

$$x'_r = x_u - \frac{\frac{f(x_u)}{2}\,(x_l - x_u)}{f(x_l) - \frac{f(x_u)}{2}}$$

Let $x_r - x'_r > 0 \rightarrow f(x_l) > 0$ *Ends up in a contradiction!*

$\Rightarrow x_r < x'_r$

i.e. **root estimate moves right** after $f(x_u) \leftarrow \frac{f(x_u)}{2}$

Bracketing **interval detection**