3. Roots of equations: bracketing methods

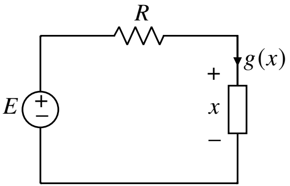
- nonlinear equation in one variable
- graphical methods
- bracketing methods
- bisection method
- false position method

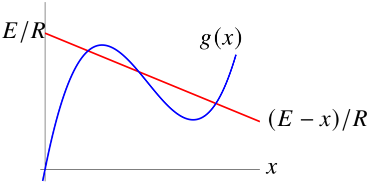
Nonlinear equation in one variable

$$f(x) = 0$$

- the *root* or *zero* is any solution of the above equation
- we assume f is a univariate continuous function on an interval $[x_l, x_u]$
- there may be one solution, multiple solutions, or no solution

Example: nonlinear resistive circuit

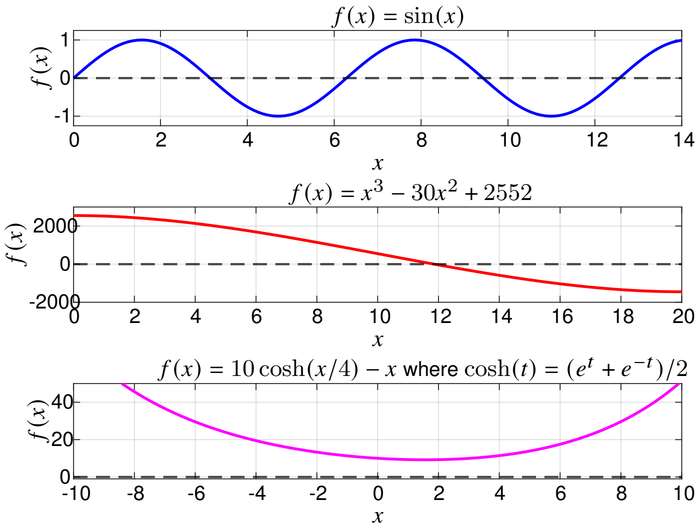




$$g(x) - \frac{E - x}{R} = 0$$

a nonlinear equation in the variable x, with three solutions

Examples


$f(x) = \sin(x)$
$f(x)$
1
0
-1
0
2
4
6
8
10
12
14
x
$f(x) = x^3 - 30x^2 + 2552$
$f(x)$
2000
0
-2000
0
2
4
6
8
10
12
14
16
18
20
x
$f(x) = 10\cosh(x/4) - x$ where $\cosh(t) = (e^t + e^{-t})/2$
$f(x)$
40
20
0
-10
-8
-6
-4
-2
0
2
4
6
8
10
x

Iterative methods

- nonlinear equations are generally difficult to solve
- obtaining a solution by finite-step algorithm is not feasible
- iterative algorithms typically start with *initial/starting point*, x_0 and compute

$$x_0, x_1, \ldots, x_i, \ldots$$

  where x_i is the i*th iterate*

- moving from x_i to x_{i+1} is called an *iteration* of the algorithm
- ideally converge to a root of the target function

$$x_i \longrightarrow x^\star \quad \text{as} \quad i \longrightarrow \infty$$

  where $f(x^\star) = 0$

Outline

- nonlinear equation in one variable
- **graphical methods**
- bracketing methods
- bisection method
- false position method

Graphical methods

- plot $f(x)$ to identify approximate root locations
- root $\approx$ where $f(x)$ crosses the x-axis
- provides **rough estimates** of roots
- estimates can be employed as starting guesses for other numerical methods
- useful to visualize:
  - function properties (multiple roots, discontinuities, ill-conditioned intervals)
  - behavior of numerical methods

Example

recall our falling parachutist equation

$$v(t) = \frac{gm}{c}(1 - e^{-(c/m)t})$$

find drag coefficient c so that $v = 40$ m/s after $t = 10$ s with $m = 68.1$ kg

- our root problem is

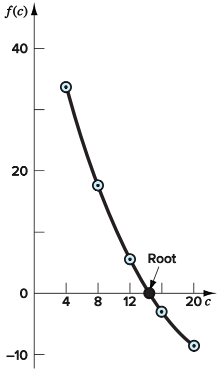
$$f(c) = \frac{9.81 \times 68.1}{c}\left(1 - e^{-(c/68.1)10}\right) - 40 = 0$$

- we evaluate $f(c)$ at trial values and plot or in MATLAB

```
% Define c range (avoid c=0 to prevent division by zero)
c = linspace(1,20,500);   % c from 1 to 200 with 500 points
% Define function
f = (9.81*68.1 ./ c) .* (1 - exp(-(c/68.1)*10))-40;
% Plot
plot(c,f,'LineWidth',2);
grid on;
```

Example

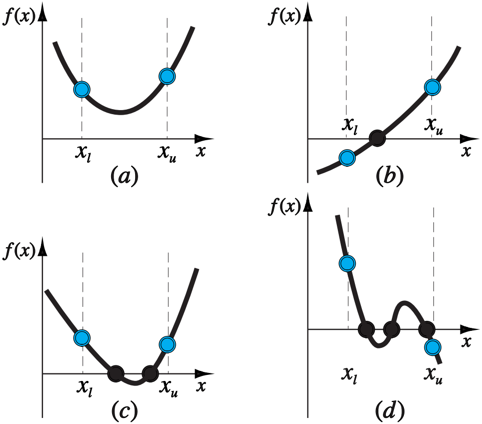
c	$f(c)$
4	34.190
8	17.712
12	6.114
16	-2.230
20	-8.368



- plot shows crossing between $c = 12$ and $c = 16$, $c^\star \approx 14.75$
- substitution check: $f(14.75) \approx 0.100$

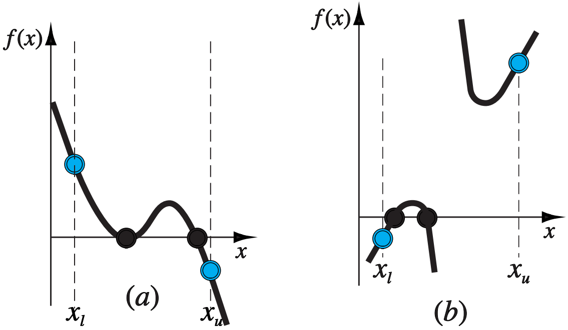
Roots in brackets

- if $f(x_l)$ and $f(x_u)$ have same sign $\Longrightarrow$ either 0 or even number of roots
- if $f(x_l)$ and $f(x_u)$ have opposite signs $\Longrightarrow$ odd number of roots in (x_l, x_u)
  - when f is real and continuous, then there is at least one root



Roots in brackets: exceptions

- *multiple roots:* function tangential to x-axis
  - *e.g.*, $f(x) = (x-2)(x-2)(x-4)$ (multiple roots)
- *discontinuous functions:* roots may not follow sign-change logic



Outline

- nonlinear equation in one variable
- graphical methods
- **bracketing methods**
- bisection method
- false position method

Bracketing methods

- many numerical methods for roots exploit a **sign change** near the root
- such approaches are called **bracketing methods**
- two initial guesses are required that lie on either side of the root
- methods reduce the bracket width systematically to converge to the solution

Incremental search

- compute $f(x)$ over interval $[x_{\min}, x_{\max}]$ with subinterval length n_s
- a root exists in the subinterval where f changes sign
- if interval length is too small, the search can be very time consuming
- if the length is too great, then a closely spaced roots might be missed

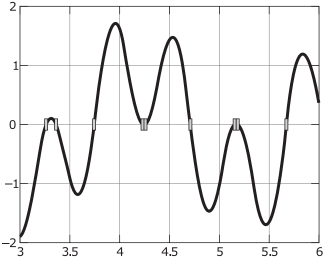
MATLAB incremental search

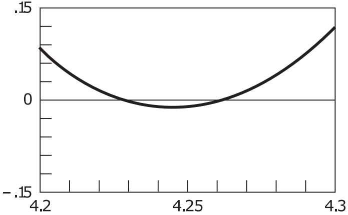
```
function xb = incsearch(func,xmin,xmax,ns)
% input:
% func = name of function
% xmin, xmax = endpoints of interval
% ns = number of subintervals
% output:
% xb(k,1) is the lower bound of the kth sign change
% xb(k,2) is the upper bound of the kth sign change
% If no brackets found, xb = [].
% Incremental search
x = linspace(xmin,xmax,ns);
f = func(x);
nb = 0; xb = []; %xb is null unless sign change detected
for k = 1:length(x)-1
if sign(f(k)) ~ = sign(f(k+1)) %check for sign change
nb = nb + 1;
xb(nb,1) = x(k);
xb(nb,2) = x(k+1);
end
end
if isempty(xb) %display that no brackets were found
disp('no brackets found')
disp('check interval or increase ns')
else
disp('number of brackets:') %display number of brackets
disp(nb)
end
```

Example

$$f(x) = \sin(10x) + \cos(3x), \quad 3 \le x \le 6$$

- initial plot suggests several roots and a possible double root near $x \approx 4.2$
- zooming on (4.2–4.3) shows *two distinct roots* near $x = 4.23$ and $x = 4.26$





Example

we locate roots between $[3, 6]$ using the previous code

- ```
  >> incsearch(@(x) sin(10*x)+cos(3*x),3,6,50)
  number of brackets:
  5
  ans =
  3.2449 3.3061
  3.3061 3.3673
  3.7347 3.7959
  4.6531 4.7143
  5.6327 5.6939
  ```
  because the subintervals are too wide, we miss possible roots at $x = 4.25$ and 5.2
- increase interval
  ```
  >> incsearch(@(x) sin(10*x) + cos(3*x),3,6,100)
  number of brackets:
  9
  ans =
  3.2424 3.2727
  3.3636 3.3939
  3.7273 3.7576
  4.2121 4.2424
  4.2424 4.2727
  4.6970 4.7273
  5.1515 5.1818
  5.1818 5.2121
  5.6667 5.6970
  ```
  ```

Outline

- nonlinear equation in one variable
- graphical methods
- bracketing methods
- **bisection method**
- false position method

Bisection method idea

- if f is real and continuous on $[x_l, x_u]$ and

$$f(x_l)\, f(x_u) < 0$$

  then there exists at least one real root in (x_l, x_u)

- *bisection (binary chopping, interval halving, Bolzano's method)*:
  - compute function value at midpoint

$$x_r = \frac{x_l + x_u}{2}$$

  - select subinterval $[x_l, x_r]$ or $[x_r, x_u]$ where sign change occurs as new interval
  - iterate by repeatedly bisecting $[x_l, x_u]$ at midpoint

- guarantees bracketing at each step

- interval width halves every iteration

Bisection method

1. start with $[x_l, x_u]$ such that $f(x_l)f(x_u) < 0$
2. compute midpoint: $x_r = (x_l + x_u)/2$ and $f(x_r)$
3. test sign:
   - if $f(x_l)f(x_r) < 0 \Rightarrow x_u = x_r$
   - else if $f(x_u)f(x_r) < 0 \Rightarrow x_l = x_r$
   - else $f(x_r) = 0$ (root found)
4. repeat until error criterion is satisfied

Example: bisection for the bungee jumper mass

- find mass m of our bungee jumper such that the velocity

$$v(t) = \sqrt{\frac{gm}{c_d}} \tanh\left(\sqrt{\frac{gc_d}{m}} t\right)$$

  equals 36 m/s after $t = 4$ s (assume $c_d = 0.25$ kg/m, $g = 9.81$ m/s^2)

- a root problem:

$$f(m) = v(4) - 36 = \sqrt{\frac{9.81m}{0.25}} \tanh\left(\sqrt{\frac{9.81 \times 0.25}{m}} \times 4\right) - 36 = 0$$

  exact value of the root is 142.7376

- use initial interval $[50, 200]$

Example: bisection for the bungee jumper mass

- iteration 1: $x_l = 50$ and $x_u = 200$,

$$x_r = \frac{50+200}{2} = 125, \quad |\varepsilon_t| = \left|\frac{142.7376 - 125}{142.7376}\right| \times 100\% = 12.43\%$$

  we have

$$f(50)f(125) = -4.579(-0.409) = 1.871$$

  so $f(125)f(150) < 0$ and root is in upper interval $[125, 150]$ ($x_l = x_r = 125$)

- iteration 2: $x_l = 125$ and $x_u = 200$,

$$x_r = \tfrac{125+200}{2} = 162.5, \quad |\varepsilon_t| = 13.85\%, \quad f(125)f(162.5) = -0.147$$

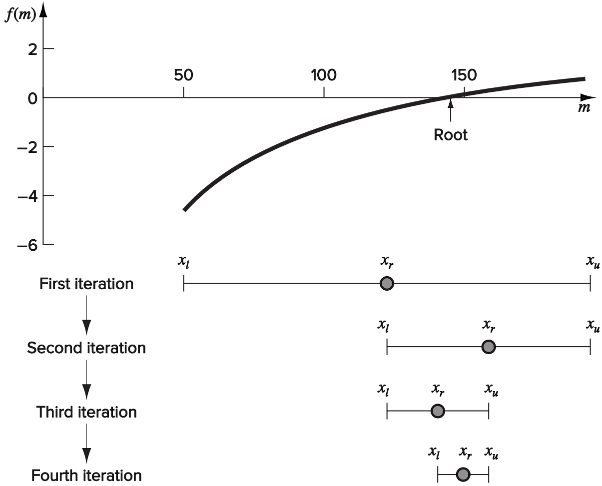
  the root is now in the lower interval $[125, 162.5]$ ($x_u = x_r = 162.5$)

- third iteration: $x_l = 125$ and $x_u = 162.5$,

$$x_r = \frac{125+162.5}{2} = 143.75, \quad |\varepsilon_t| = 0.709\%$$

- method can be repeated until the result is accurate enough to satisfy your needs

Example: bisection for the bungee jumper mass



interval width halves each iteration; root remains bracketed

Example: bisection for the parachutist drag coefficient

use bisection method to find drag coefficient c such that

$$f(c) = \frac{668.06}{c}\left(1 - e^{-0.146843c}\right) - 40 = 0$$

and initial bracket from the graph: $[12, 16]$ (true root ≈ 14.8011 for reference)

- iteration 1:

$$x_r = \tfrac{12+16}{2} = 14,\; f(12)\,f(14) = 6.114 \times 1.611 > 0 \Rightarrow \text{new bracket } [14, 16]$$

- iteration 2:

$$x_r = \tfrac{14+16}{2} = 15,\; f(14)\,f(15) = 1.611{\times}(-0.384) < 0 \Rightarrow \text{new bracket } [14, 15]$$

- iteration 3: $x_r = \frac{14+15}{2} = 14.5 \Rightarrow$ new bracket decided similarly
- ...

Termination: approximate relative error

without knowing the true root, use the approximate percent relative error

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

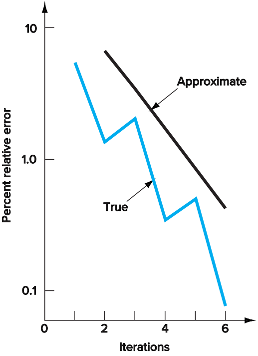
stop when $\varepsilon_a < \varepsilon_s$ (user-specified tolerance) or when iteration cap is reached

Example: continue previous example until $\varepsilon_a < 0.5\%$

iter	x_l	x_u	x_r	ε_a (%)	ε_t (%)
1	12	16	14	—	5.413
2	14	16	15	6.667	1.344
3	14	15	14.5	3.448	2.035
4	14.5	15	14.75	1.695	0.345
5	14.75	15	14.875	0.840	0.499
6	14.75	14.875	14.8125	0.422	0.077

stop at iteration 6 since $\varepsilon_a < 0.5\%$

True and approximate relative errors



- suggests that ε_a captures the general downward trend of ε_t
- ε_a is greater than ε_t
- when $\varepsilon_a < \varepsilon_s$, the computation could be terminated with confidence

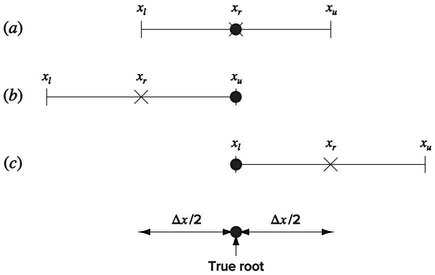
Bisection error bound

ε_a is always greater than ε_t

- approximate root is located using bisection as $x_r = \frac{x_l + x_u}{2}$
- we know that the true root lies somewhere within an interval

$$\pm\frac{x_u - x_l}{2} = \pm\frac{\Delta x}{2}$$

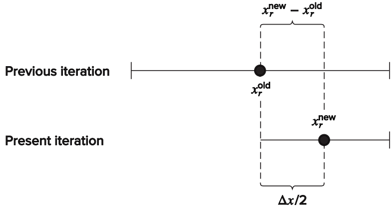
of our estimate x_r



Bisection error bound

- observe that

$$\frac{\Delta x}{2} = x_r^{\text{new}} - x_r^{\text{old}}$$



- hence, $\varepsilon_a = |\frac{x_r^{\text{new}} - x_r^{\text{old}}}{x_r^{\text{new}}}| \times 100\%$ provides an exact upper bound on the true error

Alternative approximate error expression: since

$$x_r^{\text{new}} - x_r^{\text{old}} = \frac{x_u - x_l}{2}, \qquad x_r^{\text{new}} = \frac{x_l + x_u}{2}$$

we have

$$\varepsilon_a = \left|\frac{x_r^{\text{new}} - x_r^{\text{old}}}{x_r^{\text{new}}}\right| \times 100\% = \left|\frac{x_u - x_l}{x_u + x_l}\right| \times 100\%$$

allows error estimate from the very first iteration

How many iterations do we need?

- initial interval $x_{u,0} - x_{l,0} = \Delta x_0$
- the interval is halved after each iteration and at iteration n: $x_{u,n} - x_{l,n} = \Delta x_0 / 2^n$
- after n iterations, the absolute error satisfies

$$E_a^n = |x_{r,n} - x^\star| \leq \frac{\Delta x_0}{2^n}$$

- to guarantee $E_a^n \leq E_{a,d}$, choose

$$n \;\geq\; \log_2\!\left(\frac{\Delta x_0}{E_{a,d}}\right)$$

Example: in last example with $\Delta x_0 = 16 - 12 = 4$ and $E_{a,d} = 0.0625$, we require

$$n \geq \log_2(4/0.0625) = 6$$

The bisection method

---

given: x_l, x_u with $x_l < x_u$, $f(x_l)f(x_u) < 0$, and tolerance ε_s

repeat

1. $x_r = (x_l + x_u)/2$
2. compute $f(x_r)$; **if** $f(x_r) = 0$, **return** x_r
3. **if** $f(x_r)f(x_l) < 0$, $x_u = x_r$, **else**, $x_l = x_r$
4. **stop** if $\varepsilon_a = \left|\frac{x_u - x_l}{x_u + x_l}\right| \times 100\% < \varepsilon_s$

- condition $f(x_l)f(x_u) < 0$ ensures a root exists between x_l, x_u
- x_l, x_u can be chosen from graphing the function

MATLAB implementation of bisection

```
function [root,fx,ea,iter]=bisect(func,xl,xu,es,maxit,varargin)
if nargin<3,error('at least 3 input arguments required'),end
test = func(xl,varargin{:})*func(xu,varargin{:});
if test>0,error('no sign change'),end
if nargin<4 || isempty(es), es=0.0001;end
if nargin<5 || isempty(maxit), maxit=50;end
iter = 0; xr = xl; ea = 100;
while (1)
xrold = xr; xr = (xl + xu)/2;
iter = iter + 1;
if xr ~= 0,ea = abs((xr - xrold)/xr) * 100;end
test = func(xl,varargin{:})*func(xr,varargin{:});
if test < 0
xu = xr;
elseif test > 0
xl = xr;
else
ea = 0;
end
if ea <= es || iter >= maxit,break,end
end
root = xr; fx = func(xr, varargin{:});
end
```

MATLAB implementation of bisection

we use the previous code to solve for the bungee jumper mass example:

```
 fm = @(m,cd,t,v) sqrt(9.81*m/cd)*tanh(sqrt(9.81*cd/m)*t) - v;
[mass fx ea iter] = bisect(@(m) fm(m,0.25,4,36),40,200)
mass =
142.7377
fx =
4.6089e-007
ea =
5.345e-005
iter =
21
```

Bisection: pros and cons

Pros

- guaranteed convergence if f continuous and initial bracket valid
- simple, robust, and monotonic interval reduction
- clean error bounds; iteration count predictable

Cons

- linear (slow) convergence rate
- requires bracketing; does not exploit derivative or curvature information

Outline

- nonlinear equation in one variable
- graphical methods
- bracketing methods
- bisection method
- **false position method**

False-position method

- bisection is valid but inefficient: it always divides the interval into equal halves
- *false position* (*regula falsi, linear interpolation method*) provides a more efficient alternative
- idea: use the relative magnitudes of $f(x_l)$ and $f(x_u)$ to improve the root estimate
- if $f(x_l)$ is much closer to zero than $f(x_u)$, then the root is likely closer to x_l

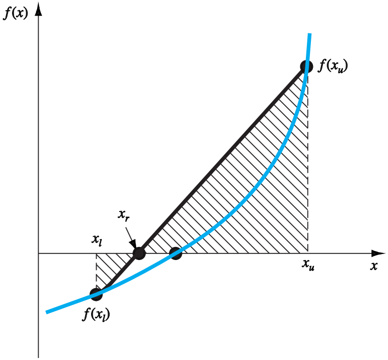
Graphical insight of false position

- instead of bisecting the interval, connect a straight line to the points

$$(x_l, f(x_l)) \quad \text{and} \quad (x_u, f(x_u))$$

- intersection of this line with the x-axis is taken as the new root estimate

- this point is called the **false position** because the curve is replaced by a line



False-position formula

using similar triangle (equating slope):

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

solving for x_r gives the *false-position* formula

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

- uses both function values and endpoints
- the interval is updated similar to bisection:
  - if $f(x_l)f(x_r) < 0$, the root lies between x_l and $x_r \Rightarrow x_u = x_r$
  - if $f(x_r)f(x_u) < 0$, the root lies between x_r and $x_u \Rightarrow x_l = x_r$
- the process repeats based on the new interval

Example: false-position on the parachutist equation

use the false-position method to determine the root of

$$f(x) = \frac{668.06}{x}\left(1 - e^{-0.146843\,x}\right) - 40$$

with initial guesses: $x_l = 12, x_u = 16$

First iteration

$$f(12) = 6.1139, \qquad f(16) = -2.2303$$

$$x_r = x_u - \frac{f(x_u)\,(x_l - x_u)}{f(x_l) - f(x_u)} = 16 - \frac{(-2.2303)(12-16)}{6.1139 - (-2.2303)} = 14.309$$

true relative error $\approx 0.88\%$ (for reference)

since $f(x_l)f(x_r) < 0$, the new bracket is $[x_l, x_u] = [12,\ 14.309]$ (*i.e.*, $x_u = x_r$)

Example: false-position on the parachutist equation

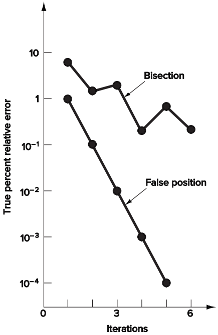
Second iteration

$$x_l = 12, \quad f(x_l) = 6.1139, \qquad x_u = 14.9309, \quad f(x_u) = -0.2515$$

$$x_r \;=\; 14.9309 - \frac{(-0.2515)(12 - 14.9309)}{6.1139 - (-0.2515)} \;=\; 14.8151$$

true and approximate relative errors: $\varepsilon_t \approx 0.09\%,\ \varepsilon_a \approx 0.78\%$

further iterations refine the estimate similarly



False position versus bisection

- false position can decrease true error *faster* than bisection
  - more informative placement of x_r
- unlike bisection, the interval *need not shrink symmetrically*
  - one endpoint can remain fixed while the other approaches the root
- consequence: the interval width is *not* a reliable error bound for false position
- using $\varepsilon_a = \left|\frac{x_r^{\text{new}} - x_r^{\text{old}}}{x_r^{\text{new}}}\right| 100\%$ is conservative when convergence is rapid: numerator largely reflects the previous iteration's discrepancy

Example: pitfalls of false position

locate the root of $f(x) = x^{10} - 1$ on $[0,\ 1.3]$ using bisection and false-position

true root $x = 1$

Bisection

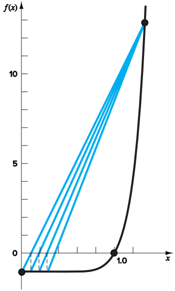
iter	x_l	x_u	x_r	$\varepsilon_a(\%)$	$\varepsilon_t(\%)$
1	0	1.3	0.65	100.0	35
2	0.65	1.3	0.975	33.3	2.5
3	0.975	1.3	1.1375	14.3	13.8
4	0.975	1.1375	1.05625	7.7	5.6
5	0.975	1.05625	1.015625	4.0	1.6

after 5 iterations, $\varepsilon_t < 2\%$

Example: pitfalls of false position

False position

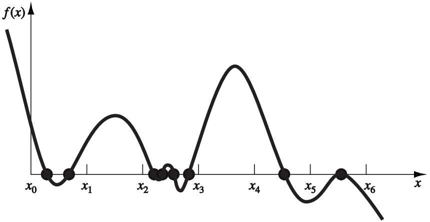
iter	x_l	x_u	x_r	$\varepsilon_a\,(\%)$	$\varepsilon_t\,(\%)$
1	0	1.3	0.09430	—	90.6
2	0.09430	1.3	0.18176	48.1	81.8
3	0.18176	1.3	0.26287	30.9	73.7
4	0.26287	1.3	0.33811	22.3	66.2
5	0.33811	1.3	0.40788	17.1	59.2



- very slow progress; also note cases with $\varepsilon_a < \varepsilon_t$ (misleading)
- interpretation: function shape violates: “closer f-value $\Rightarrow$ closer to root”
- **one-sidedness:** one endpoint often remains fixed while the other moves, causing poor convergence with strong curvature

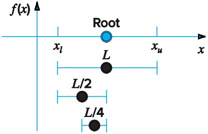
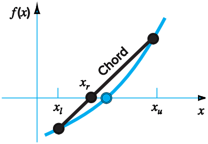
Checking for all roots

beyond verifying a single root, ensure *all possible roots* are located



- **incremental search:**
  - evaluate $f(x)$ at small increments across region
  - sign change $\Rightarrow$ root in that subinterval
  - endpoints serve as initial guesses for bracketing methods
- always supplement with:
  - function plots (plotting $f(x)$ is a useful first step)
  - insight from physical meaning of the problem

Summary

Method	Formulation	Graphical Interpretation	Errors and Stopping Criteria
Bisection	$x_r = \dfrac{x_l + x_u}{2}$ If $f(x_l)f(x_r) < 0,\ x_u = x_r$ $f(x_l)f(x_r) > 0,\ x_l = x_r$	Bracketing methods: 	Stopping criterion: $\left\lvert \dfrac{x_r^{new} - x_r^{old}}{x_r^{new}} \right\rvert 100\% \le \epsilon_s$
False position	$x_r = x_u - \dfrac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)}$ If $f(x_l)f(x_r) < 0,\ x_u = x_r$ $f(x_l)f(x_r) > 0,\ x_l = x_r$		Stopping criterion: $\left\lvert \dfrac{x_r^{new} - x_r^{old}}{x_r^{new}} \right\rvert 100\% \le \epsilon_s$

References and further readings

- S. C. Chapra and R. P. Canale. *Numerical Methods for Engineers* (8th edition). McGraw Hill, 2021. (Ch.5)
- S. C. Chapra. *Applied Numerical Methods with MATLAB for Engineers and Scientists* (5th edition). McGraw Hill, 2023. (Ch.5)