Roots: Bracketing Methods

Berlin Chen
Department of Computer Science & Information Engineering
National Taiwan Normal University

Reference:

1. *Applied Numerical Methods with MATLAB for Engineers*, Chapter 5 & Teaching material

Chapter Objectives (1/2)

- Understanding what roots problems are and where they occur in engineering and science
- Knowing how to determine a root graphically
- Understanding the incremental search method and its shortcomings

$$ax^2 + bx + c = 0 \quad \Rightarrow \quad x = \frac{-b \mp \sqrt{b^2 - 4ac}}{2a}$$

$$ax^5 + bx^4 + cx^3 + dx^2 + ex + f = 0 \quad \Rightarrow x = ?$$

$$\sin x + x = 0 \quad \Rightarrow x = ?$$

Chapter Objectives (2/2)

- Knowing how to solve a roots problem with the bisection method
- Knowing how to estimate the error of bisection and why it differs from error estimates for other types of root location algorithms
- Understanding false position and how it differs from bisection

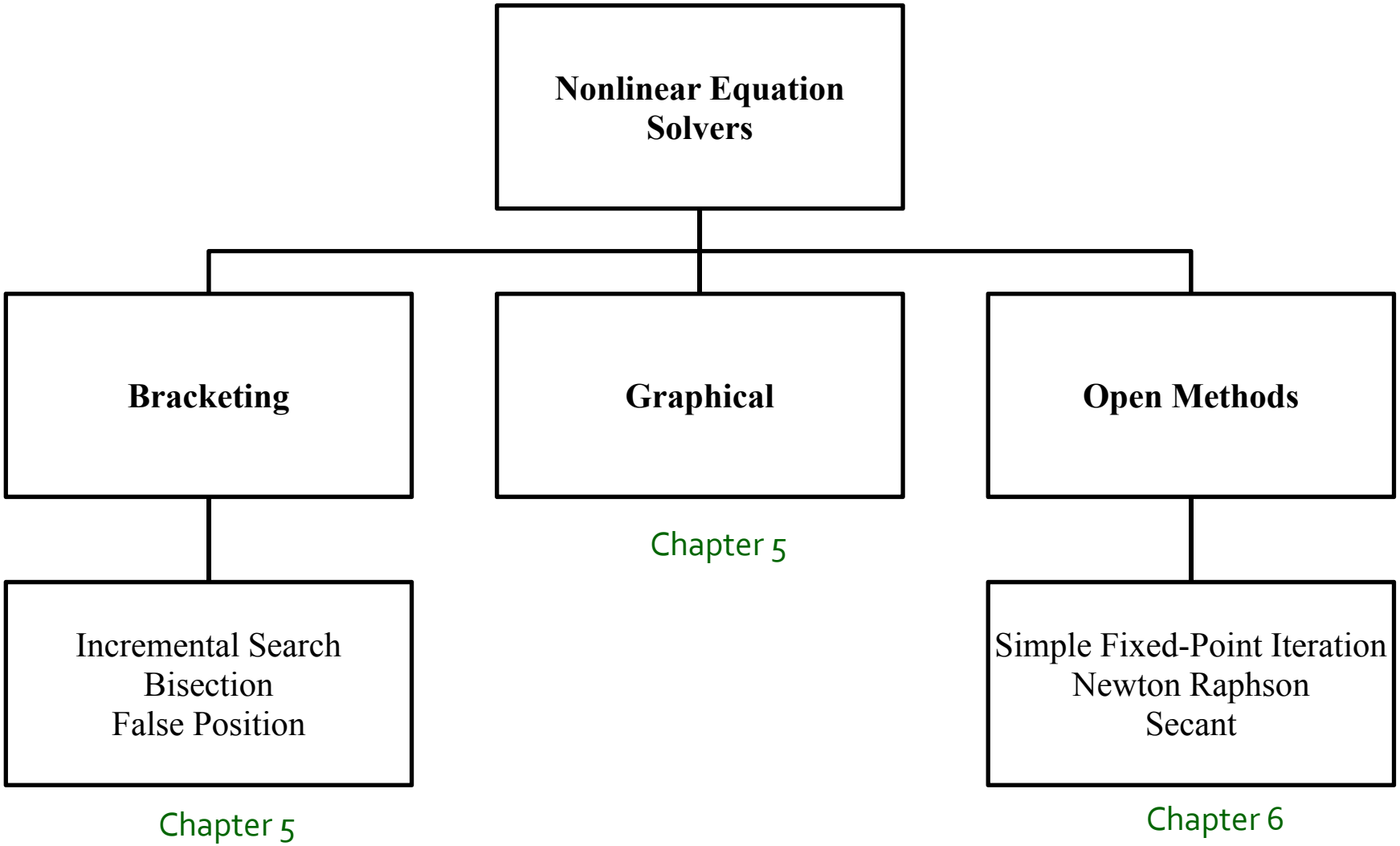
Roots

- "Roots" problems occur when some function *f* can be written in terms of one or more dependent variables *x*, where the solutions to *f(x)=0* yields the solution to the problem
- These problems often occur when a design problem presents an implicit equation for a required parameter

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

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

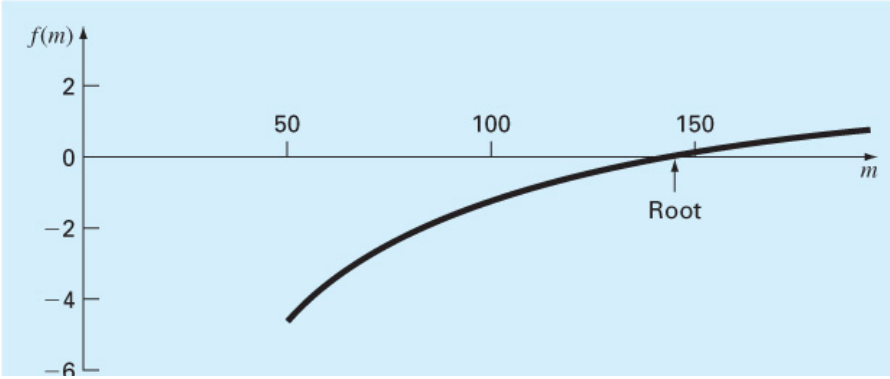
Taxonomy of Root-finding Methods



– We can also employ a hybrid approach (**Bracketing + Open Methods**)

Graphical Methods (1/2)

- A simple method for obtaining the estimate of the root of the equation *f(x)*=0 is to make a plot of the function and observe where it crosses the *x*-axis

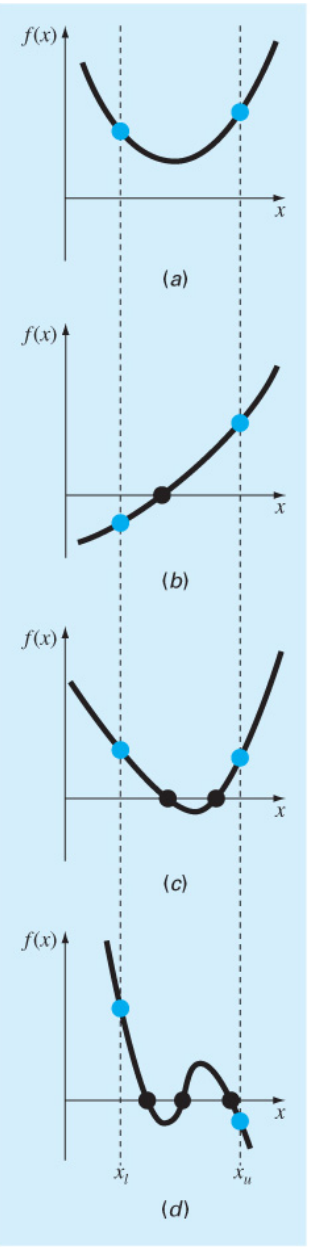


$$f(m)=\sqrt{\frac{gm}{c_d}}\tanh\left(\sqrt{\frac{gc_d}{m}}4\right)-36$$

- Graphing the function can also indicate where roots may be and where some root-finding methods may fail
- The estimate of graphical methods (an rough estimate) can be employed as starting guesses for other numerical methods

Graphical Methods (1/2)

- Therefore, graphical interpretation of the problem are useful for understanding the properties of the functions and anticipating the pitfalls of the numerical methods



a) Same sign, no roots
b) Different sign, one root
c) Same sign, two roots
d) Different sign, three roots

Bracketing Methods

- *Bracketing methods* are based on making two initial guesses that "bracket" the root - that is, are on either side of the root
- Brackets are formed by finding two guesses x_l and x_u where the sign of the function changes; that is, where $f(x_l)\, f(x_u) < 0$

> In general, if $f(x)$ is a real and continuous in the interval from x_l to x_u and $f(x_l)$ and $f(x_u)$ have opposite signs, that is
>
> $$f(x_l)\ f(x_u) < 0$$
>
> then there is at least one real root between x_l and x_u.

  - We can use the ***incremental search*** method (an automatic approach to obtain initial guesses) tests the value of the function at evenly spaced intervals and finds brackets by identifying function sign changes between neighboring points

Incremental Search Hazards

- If the spacing between the points of an incremental search are too far apart, brackets may be missed due to capturing an even number of roots within two points
- Incremental searches cannot find brackets containing even-multiplicity roots regardless of spacing

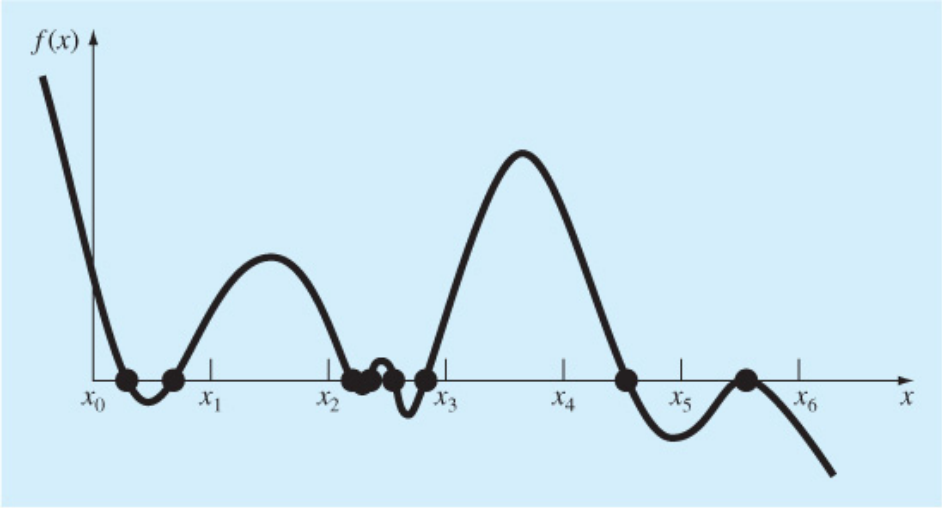


FIGURE 5.3
Cases where roots could be missed because the incremental length of the search procedure is too large. Note that the last root on the right is multiple and would be missed regardless of the increment length.

Bisection

- The *bisection method* is a variation of the incremental search method in which the interval is always divided in half
- If a function changes sign over an interval, the function value at the midpoint is evaluated
- The location of the root is then determined as lying within the subinterval where the sign change occurs
- The absolute error is reduced by a factor of 2 for each iteration

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

where $t=4, v(t)=36$

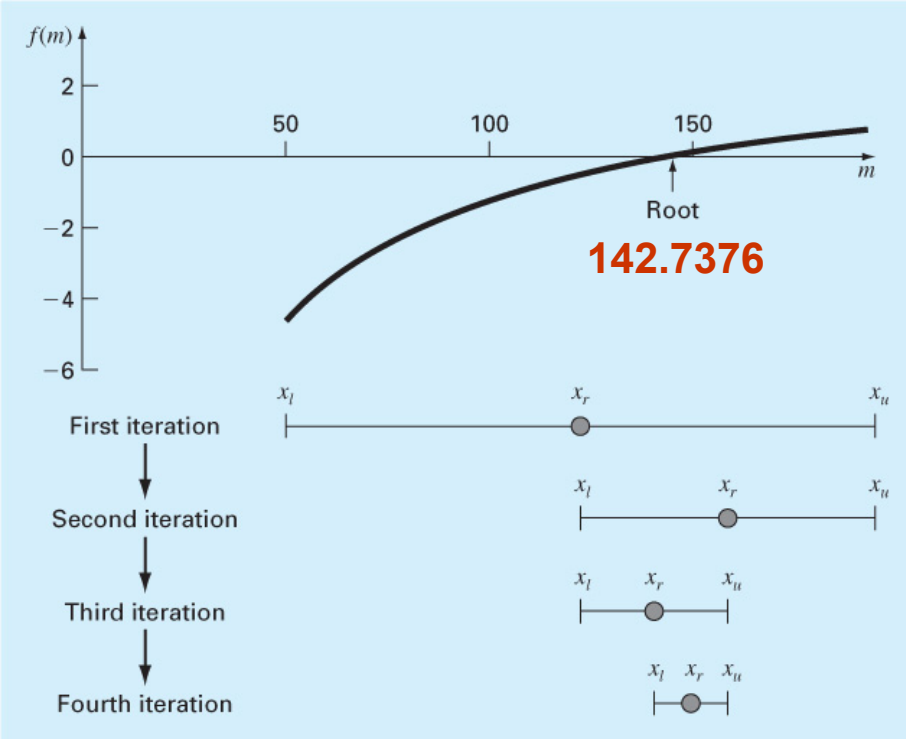


FIGURE 5.5
A graphical depiction of the bisection method. This plot corresponds to the first four iterations from Example 5.3.

Bisection: An Example

[First iteration]

$$x_r = \frac{50+200}{2} = 125$$

$$|\varepsilon_t| = \left|\frac{142.7376-125}{142.7376}\right| \times 100\% = 12.43\%$$

$$\because f(50)f(125) = -4.579 \times (-0.409) = 1.871$$

$$\therefore \text{the new low bound will be } 125$$

[Second iteration]

$$x_r = \frac{125+200}{2} = 162.5$$

$$|\varepsilon_t| = \left|\frac{142.7376-162.5}{142.7376}\right| \times 100\% = 13.85\%$$

$$\because f(125)f(162.5) = -0.409 \times (0.359) = -0.147$$

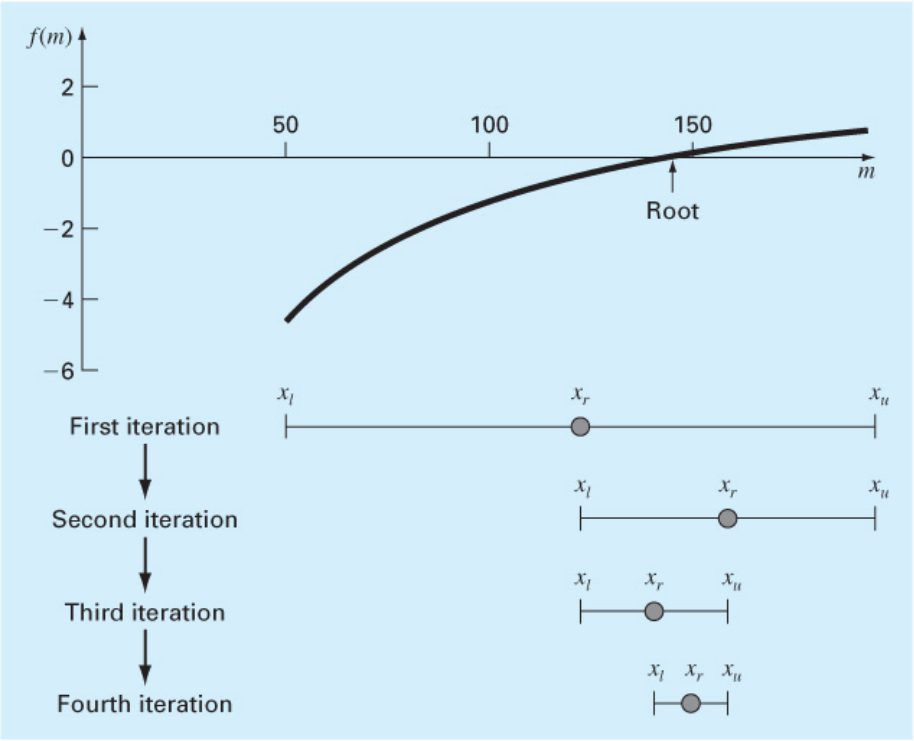
$$\therefore \text{the new upper bound will be } 162.5$$

[Third iteration]

$$x_r = \frac{125+162.5}{2} = 143.75$$

$$|\varepsilon_t| = \left|\frac{142.7376-143.75}{142.7376}\right| \times 100\% = 0.709\%$$

$$\vdots$$



Programming Bisection

```
function [root,ea,iter]=bisect(func,xl,xu,es,maxit,varargin)
% bisect: root location zeroes
%   [root,ea,iter]=bisect(func,xl,xu,es,maxit,p1,p2,...):
%      uses bisection method to find the root of func
% input:
%   func = function handle
%   xl, xu = lower and upper guesses
%   es = desired relative error (default = 0.0001%)
%   maxit = maximum allowable iterations (default = 50)
%   p1,p2,... = additional parameters used by func
% output:
%   root = real root
%   ea = approximate relative error (%)
%   iter = number of iterations

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;
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;
```

Error Estimate

- For the bisection method, we might decide that we should terminate when the error drops below, say, 0.5%
  - However, in reality, we do not have knowledge of the true root!

- One way to get rid of the need of knowledge of the true root is to estimate **an approximate percent relative error** defined as follows

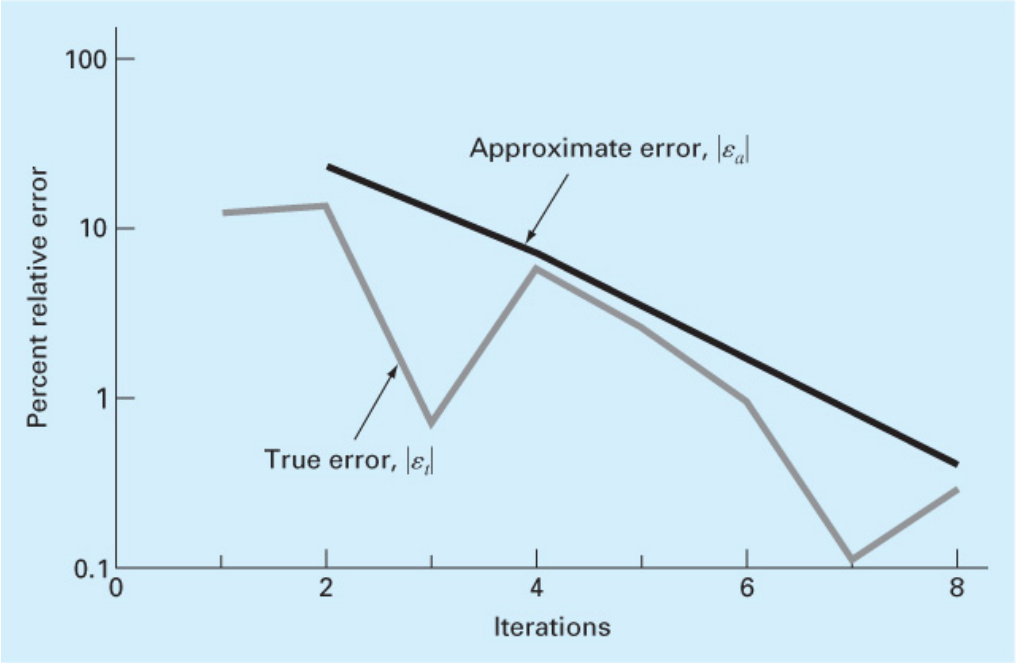
$$\left|\varepsilon_a\right| = \left|\frac{x_r^{new} - x_r^{old}}{x_r^{new}}\right| \times 100\%$$

Errors of Bisection (1/2)

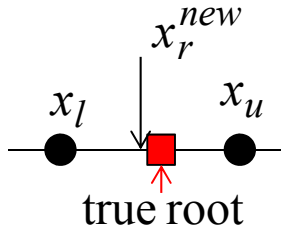
$$f(m)=\sqrt{\frac{gm}{c_d}}\tanh\left(\sqrt{\frac{gc_d}{m}}t\right)-v(t)$$

where $t=4, v(t)=36$

$|\varepsilon_a|$ captures the general downward trend of $|\varepsilon_t|$



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



– The true ($|\varepsilon_t|$) and approximate ($|\varepsilon_a|$) errors are far apart when the interval happen to be centered on the true root. They are close when the true root falls at either end of the interval

Bisection Errors (2/2)

- The absolute error of the bisection method is solely dependent on the absolute error at the start of the process (the space between the two guesses) and the number of iterations:

$$E_a^n = \frac{\Delta x^0}{2^n} \quad \text{where } \Delta x^0 = x_u^0 - x_l^0$$

- The required number of iterations to obtain a particular absolute error can be calculated based on the initial guesses:

$$n = \log_2\left(\frac{\Delta x^0}{E_{a,d}}\right) \quad \text{where } E_{a,d} \text{ is the desired error}$$

- The neatness of the error analysis of the bisection method is a positive feature that makes it attractive for many applications

False Position

- The ***false position*** method is another bracketing method
  - Also called the ***linear interpolation*** method
- It determines the next guess not by splitting the bracket in half but by **connecting the endpoints with a straight line and determining the location of the intercept of the straight line (x_r)**
- The value of x_r then replaces whichever of the two initial guesses yields a function value with the same sign as $f(x_r)$

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

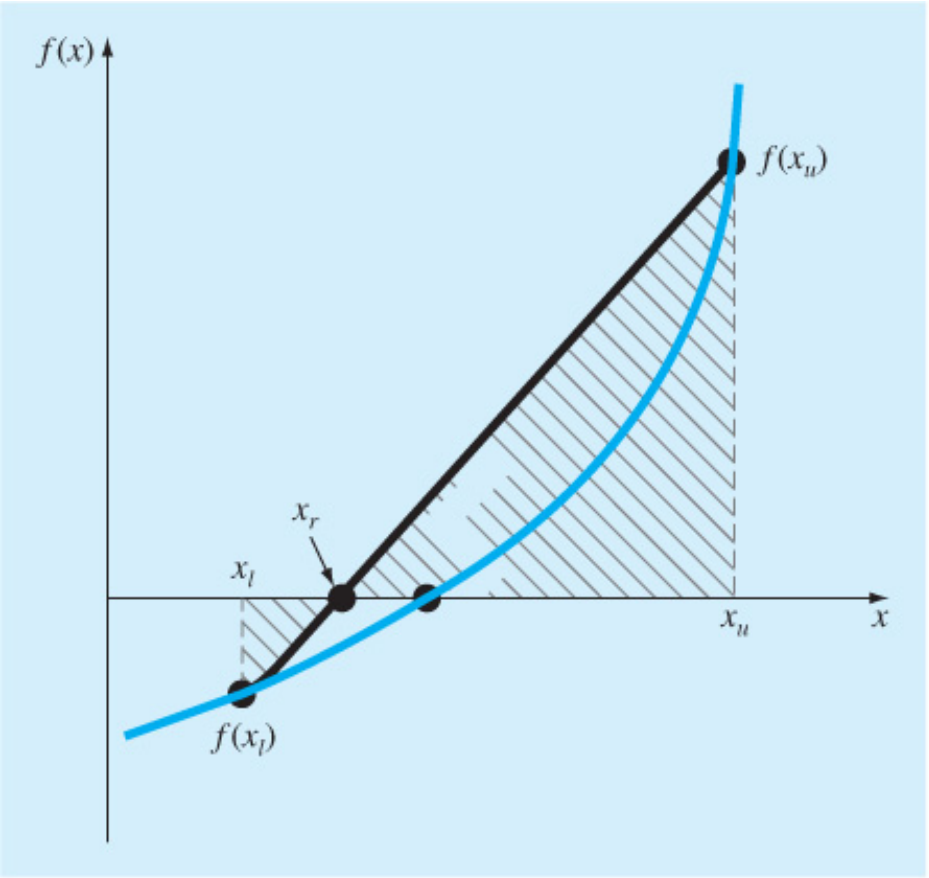
False Position: An Illustration

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

$$\frac{f(x_l) - f(x_u)}{x_l - x_u} = \frac{0 - f(x_u)}{x_r - x_u}$$

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

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



Implication from the plot

If $f(x_l)$ is much closer to zero than $f(x_u)$, then the root should be much closer to x_l than x_u? (not always the truth)

Bisection vs. False Position

- Bisection does not take into account the shape of the function; this can be good or bad depending on the function!
- A “bad” case for False Position:

$$f(x) = x^{10} - 1$$

Solution. Using bisection, the results can be summarized as

Iteration	x_l	x_u	x_r	ε_a (%)	ε_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

Thus, after five iterations, the true error is reduced to less than 2%. For false position, a very different outcome is obtained:

Iteration	x_l	x_u	x_r	ε_a (%)	ε_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

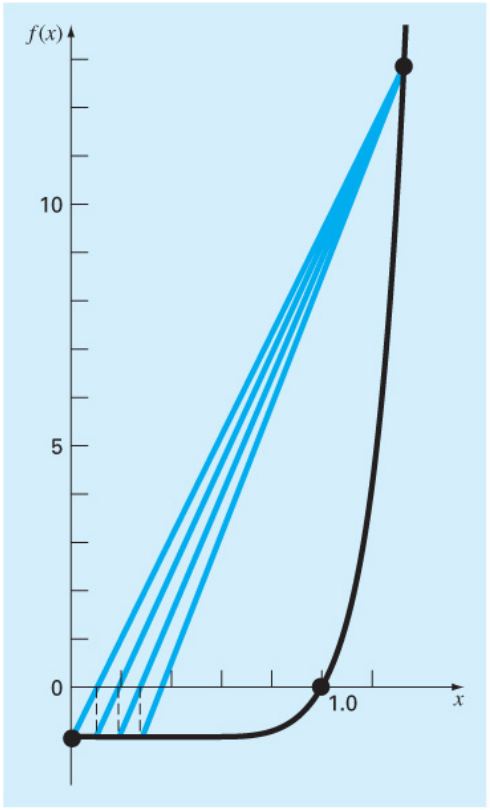


FIGURE 5.9
Plot of $f(x) = x^{10} - 1$, illustrating slow convergence of the false-position method.

Discussions

- Blanket generalizations regarding root-location methods are usually not possible
  - There are invariably cases violate our conclusion/implication such as "*false position is superior to bisection*"

- In addition to estimating an approximate percent relative error $|\varepsilon_a| = \left| \frac{x_r^{new} - x_r^{old}}{x_r^{new}} \right| \times 100\%$, our result $f\left(x_r^{new}\right)$ should always be checked by substituting the root estimate into the original equation and determining where the result $f\left(x_r^{new}\right)$ is close to zero