Part 2

Chapter 5

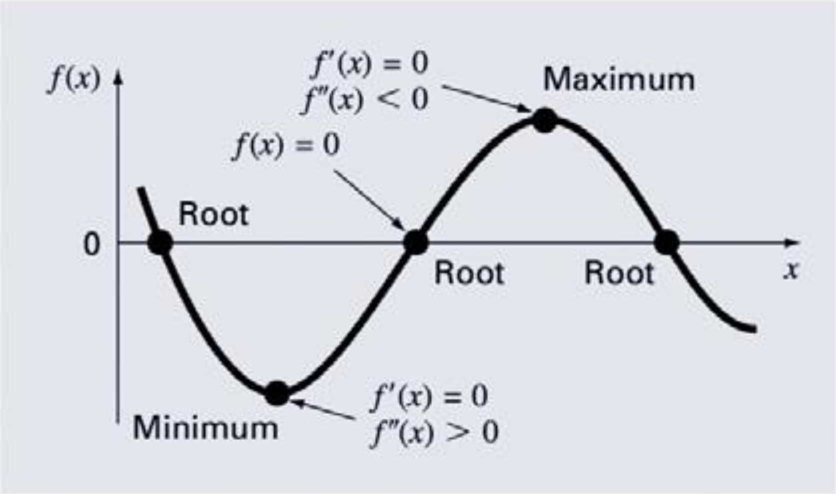
Roots: Bracketing Methods

Overview of Part 2

- To find the roots of general second order polynomial, the quadratic formula is used

$$f(x) = ax^2 + bx + c = 0 \qquad x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- There are many other functions where the formula for finding roots is not available.
  → Approximate solution technique to find roots of f(x)=0.
  → Bracketing methods (chap. 5) and open methods (chap. 6)



- Beside roots, it is often times required to find maximum and minimum values of functions, which process is referred to as optimization (chap. 7)

Chapter Objectives

- 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.
- 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 method and how it differs from bisection.

Introduction

- A bungee jumper's chances of sustaining a significant injury increase significantly, if the free fall velocity exceeds 36 m/s after 4 sec of freefall.
- Find the maximum weight for jumper who does not match this criterion (The drag coefficient is 0.25 kg/m)
  → You can't solve the equation explicitly for m.

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

- Alternative way of solving this problem is to move the v term to the left and arrange the equation in the form of *f(x)=0*.

  → The answer is the value of x that makes the function f equal to zero. → roots problem.

$$f(m) = \sqrt{\frac{gm}{c_d}} \tanh\left(\sqrt{\frac{gc_d}{m}} t\right) - v(t) = 0$$

m is said to be implicit

Roots

- "Roots" problems, f(x)=0, occur when some function f can be written in terms of one or more independent variables x,
- These problems often occur when a design problem presents an implicit equation for a required parameter.
- As for explicit equation, the computation can be done simply and quickly

Example 5.1 Graphical approach

- Determine the mass of the bungee jumper with a drag coefficient of 0.25kg/m to have a velocity of 36m/s after 4 s of the free fall.

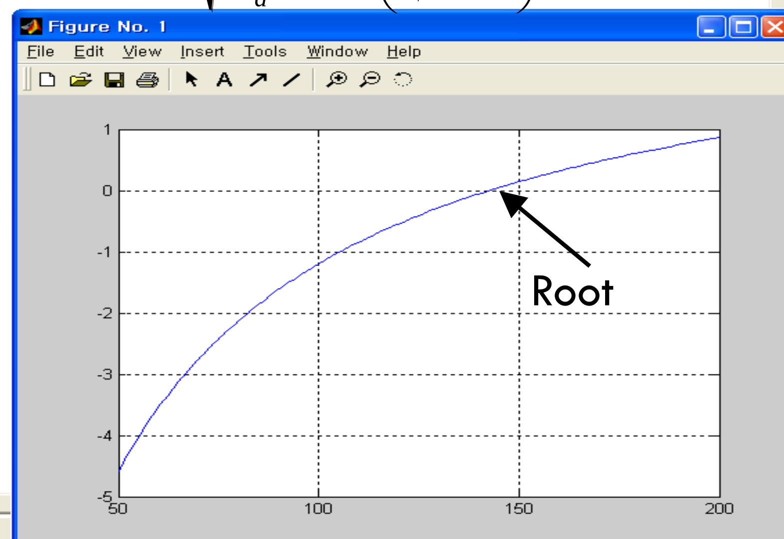
Command Window

```
>> cd = 0.25; g = 9.81; v = 36; t = 4;
>> mp = linspace (50, 200);
>> fp = sqrt(g*mp/cd).*tanh(sqrt(g*cd./mp)*t) - v;
>> plot(mp,fp), grid

>> sqrt(g*145/cd)*tanh(sqrt
    (g*cd/145)*t)-v
ans =
    0.0456
>>sqrt(g*145/cd)*tanh(sqrt
    (g*cd/145)*t)
ans =
    36.0456
```

Ready

$$f(m)=\sqrt{\frac{gm}{c_d}}\tanh\left(\sqrt{\frac{gc_d}{m}}t\right)-v(t)=0$$



Graphical approach

- 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.

- Graphing the function can also indicate where roots may be and where some root-finding methods may fail:

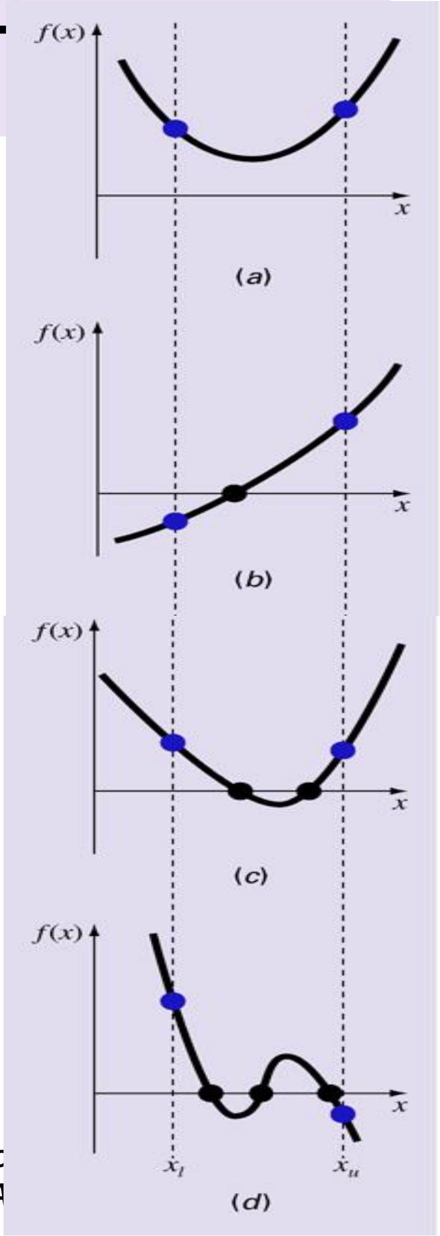
a) Same sign, no roots

b) Different sign, one root

c) Same sign, two roots

d) Different sign, three roots

- This method is for obtaining rough estimates of roots, not for precise ones.
  - Starting guesses for numerical methods
  - Understanding the properties of the functions.
  - Predicting the pitfalls of the numerical methods

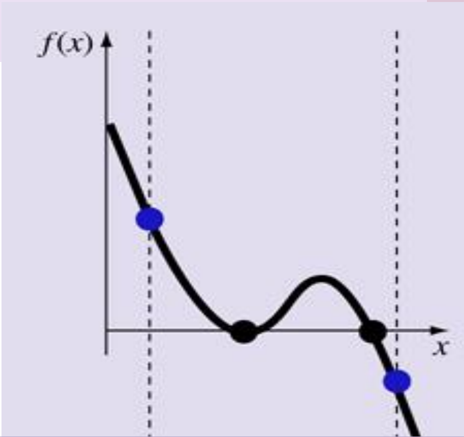


Graphical Methods (cont.)

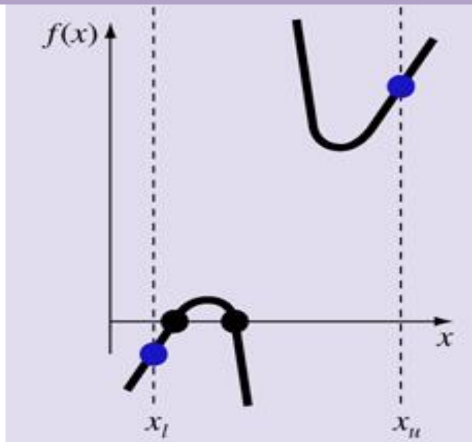
- $f(x_l)\ f(x_u) < 0$
  where a lower bound is x_l and an upper bound is x_u.
  That is, the sign of the function changes
  → Generally odd number of roots within the interval.

- $f(x_l)\ f(x_u) > 0$
  That is, the sign of the function does not change
  → Generally even number (including zero) of roots within the interval.

- Exception: graphical method helps in this case



Multiple roots when function is tangential to the x axis



Discontinuous functions

Bracketing Methods

- Trial and error methods : Require initial guesses.

  - Bracketing method and open method

- Bracketing methods are based on making two initial guesses that "bracket" the root - that is, are on either side of the root.

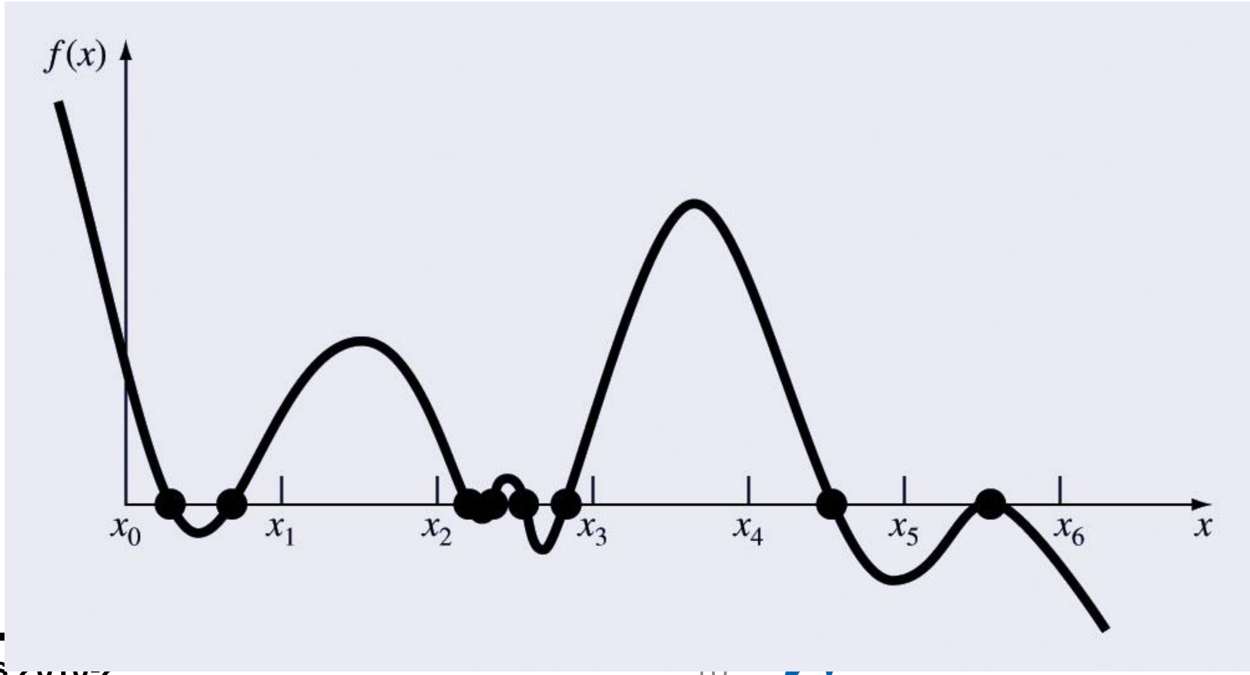
- Brackets are formed by finding two guesses x_l and x_u where the sign of the function changes; that is, where $f(x_l)\, f(x_u) < 0$

  → There is at least one real root between x_l and x_u

- The incremental search method 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.
- If the spacing is too small, the search can be very time consuming.



M-file to implement an incremental search (1)

```
function xb = incsearch(func, xmin, xmax, ns)
%  finds brackets of x that contain sign changes of
%  a function on an interval
% input:
%  func= name of function
%  xmin, xmax = endpoints of interval
%  ns = (optional) number of subintervals along x
% output:
%  xb(k,1) is the lower bound of the kth sign changes
%  xb(k,2) is the upper bound of the kth sign changes
%  If no brackets found, kb =[].
```

M-file to implement an incremental search (2)

```
if nargin <4, ns =50; end    % if ns blank set to 50
% Incremental search
x = linspace(xmin, xmax, ns);
f =  feval(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
```

M-file to implement an incremental search (3)

```
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 5.2 (1)

$$f(x) = \sin(10x) + \cos(3x) = 0$$

Find the roots of f(x)

Command Window

File Edit View Web Window Help

```
>> incsearch( @x sin(10*x)+cos(3*x), 3, 6)
nb =
     0
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
```

Ready

Example 5.2 (2)

Command Window

File Edit View Web Window Help

```
>>

>> incsearch(@x sin(10*x)+cos(3*x),3,6, 100)
nb =
     0
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
```

Ready

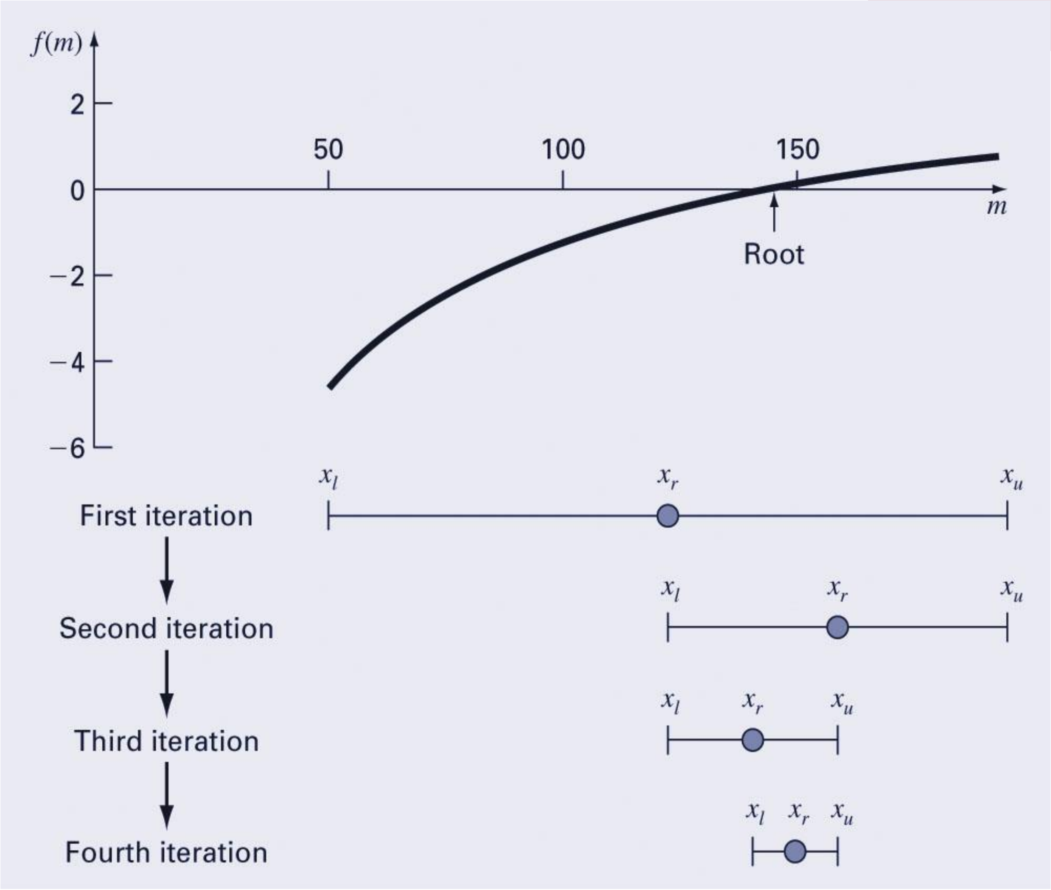
Bisection

- The bisection method is a variation of the incremental search method in which the interval is always divided in half.

- If $f(x_l)\,f(x_r) > 0$ then x_r turns into x_l

- If $f(x_l)\,f(x_r) < 0$ then x_r turns into x_u

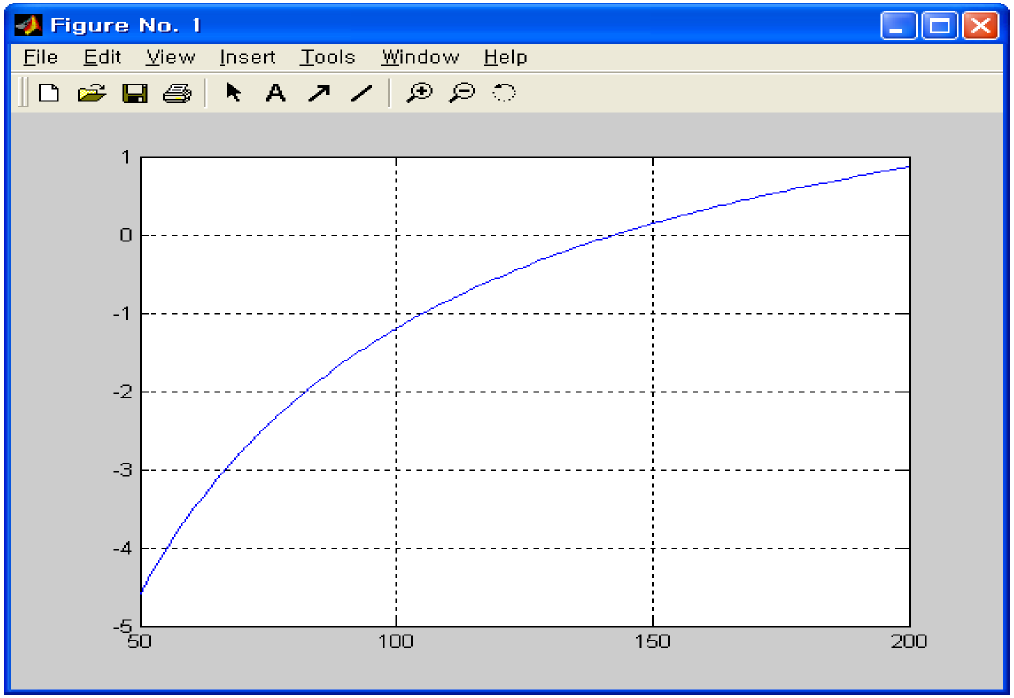
- The absolute error is reduced by a factor of 2 for each iteration.



School of Mechanical Engineering
Chung-Ang University

Example 5.4

- Use bisection method to solve the same problem approached graphically in Example 5.1 until the approximate error falls below stopping criterion of $\varepsilon_s = 0.5\%$.



$$f(m) = \sqrt{\frac{gm}{c_d}} \tanh\left(\sqrt{\frac{gc_d}{m}} t\right) - v(t) = 0$$

approximate error

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

Example 5.4

True percent relative error

$$|\varepsilon_t| = \left|\frac{x^{true} - x_r^{old}}{x^{true}}\right| 100\% \qquad x^{true} = 142.7376$$

Iteration	x_l	x_u	x_r	$\lvert\varepsilon_a\rvert$	$\lvert\varepsilon_t\rvert$
1	50	200	$\frac{50+200}{2} = 125$	N/A	$\left\lvert\frac{142.7376-125}{142.7376}\right\rvert 100\% = 12.43\%$
2	125	200	$\frac{125+200}{2} = 162.5$	23.08	13.85
3	125	162.5	143.75	13.04	0.71
4	125	143.75	134.375	6.98	5.86
5	134.375	143.75	139.0625	3.37	2.58
6	139.0625	143.75	141.4063	1.66	0.93
7	141.4063	143.75	142.5781	0.82	0.11
8	142.5781	143.75	143.1641	0.41	0.30

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 = name of function
%   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;
```

Bisection Error

- The absolute error after n th iternation by 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:

$$E_a^0 = x_u^0 - x_l^0, \quad E_a^1 = \frac{x_u^0 - x_l^0 = \Delta x^0}{2} \quad \Rightarrow \quad E_a^n = \frac{\Delta x^0}{2^n}$$

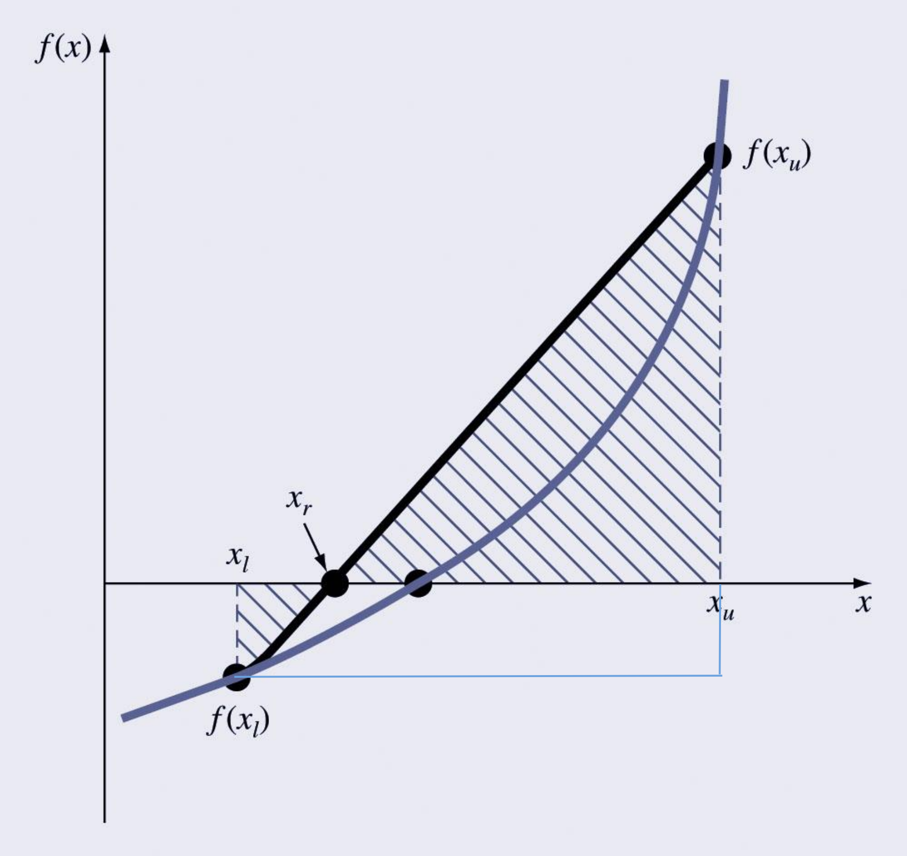
- The required number of iterations to obtain a particular absolute error can be calculated based on the initial guesses:

$$n = \log_2\left(\frac{\Delta x^0}{E_{a,d}}\right)$$

$E_{a,d}$: Desired error

False Position (1)

- The false position method is another bracketing method and it is very similar to bisection 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).



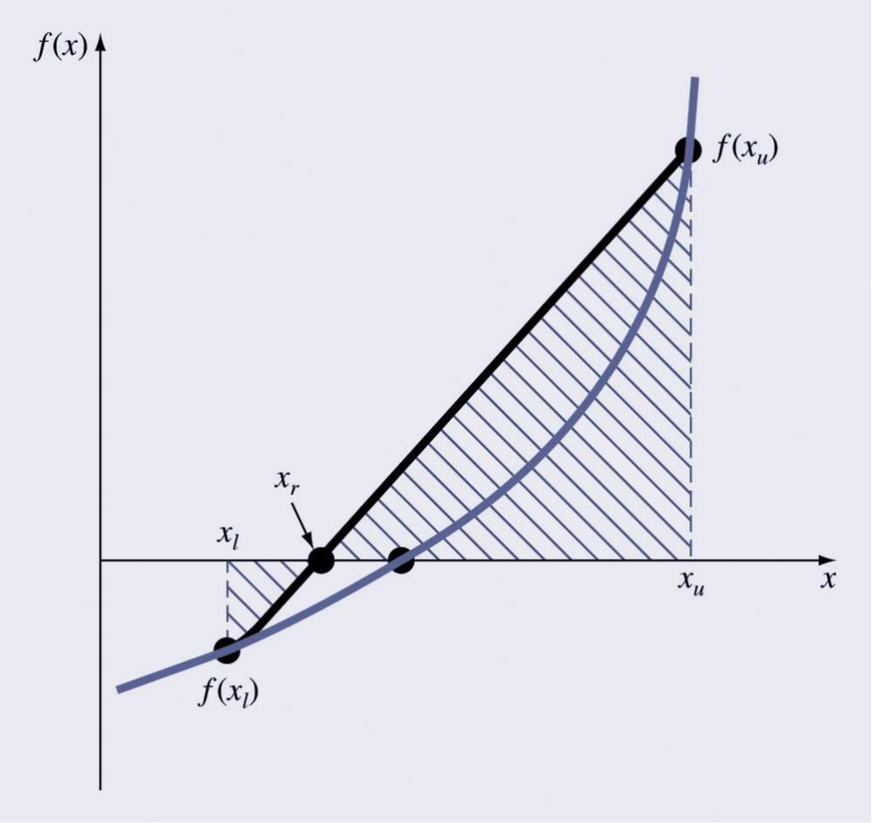
$$x_r = x_u - \frac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)}$$

False Position (2)

- The value of x_r then replaces whichever of the two initial guesses yields a function value with the same sign as $f(x_r)$.

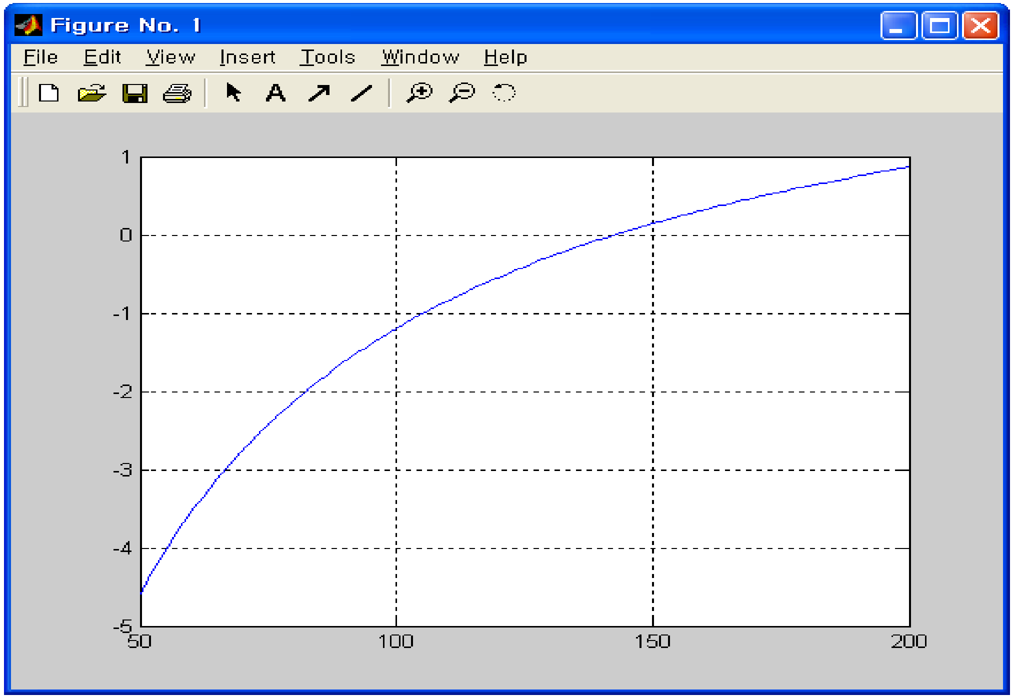
- If $f(x_l)\, f(x_r) > 0$ then x_r turns into x_l

- If $f(x_l)\, f(x_r) < 0$ then x_r turns into x_u



Example 5.5

- Use false position method to solve the same problem approached graphically in Example 5.1 until the approximate error falls below stopping criterion of $\varepsilon_s = 0.5\%$.



$$f(m) = \sqrt{\frac{gm}{c_d}} \tanh\left(\sqrt{\frac{gc_d}{m}}t\right) - v(t) = 0$$

approximate error

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

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

Bisection

n	x_l	x_u	x_r	ε_a (%)	ε_t (%)
1	0	1.3	0.65	100.0	35.0
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

False position

n	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