5. Roots: Bracketing Methods

HoHee Kim

Roots(근): f(x)=0 의 해(solution) 즉, f(x)가 x축과 만날 때의 x

f(x) = $ax^2+bx+c = 0$ → $x = \dfrac{-b \pm \sqrt{b^2-4ac}}{2a}$ 쉽게 구하지만,

근을 쉽게 구하지 못하는 함수들은 수치해법으로 근을 구함

☞ 그래프 같은 방법으로 대략 추정근을 찾아서 시행착오를 반복하여 찾음

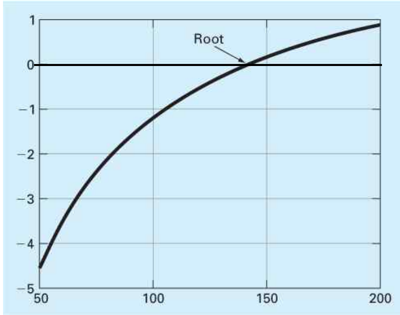
- **Bracketing Methods** : 구간이 정해지므로 2개의 초기값 필요
  확실하게 근을 찾으나 속도느림
  ① **Bisection**
  ② **False position**
  5 장
- **Open Methods** : 구간은 불필요하지만 1개 이상의 초기값 필요
  속도는 빠르지만 발산할 가능성 있음
  ① **Simple fixed-point iteration**
  ② **Newton-Raphson Method**
  ③ **Secant Method**
  6 장
- Bracketing + Open Methods ⇒ Brent's Method 6 장

Graphical Methods : 대략 추정 값을 얻는 데 이용, 정확성 결여

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

t=4일 때 속도가 36m/s 인 사람의 질량은?

(c_d = 0.25 kg/m, g = 9.81 m/s^2)



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

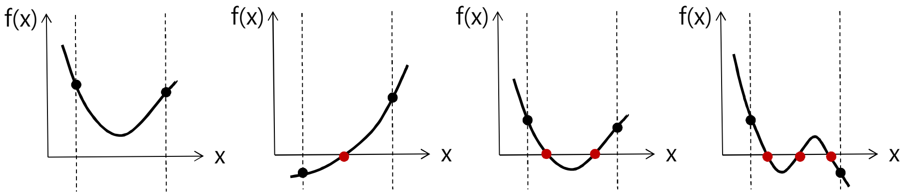
```
>> cd=0.25; g=9.81; v=36; t=4;
>> mp = linspace(50,200);
>> fp = sqrt(g*mp/cd).*tanh(sqrt(g*cd./mp)*t)-v ;
>> plot(mp, fp), grid
```

☞ 위의 그래프가 나타남

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

m=145 일 때

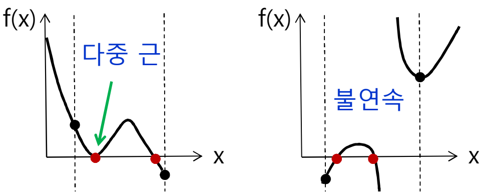
Bracketing Methods & Initial Guesses : 구간의 경계치 설정 필요



(a) (b) (c) (d)

a), c) 경계에서의 함수 값의 부호가 같을 때 근이 없거나 짝수 개,

b), d) 〃 부호가 다를 때 근이 하나이거나 홀수 개



← 아닌 경우도 있음

☞ 모든 경우를 수용하는 알고리즘 개발은 어렵다

Incremental Search: 함수 부호 이용해 근이 있는 구간을 찾아내는 방법

- 함수 f(x) 가 두 추정 값 x_l 와 x_u 사이에서 실수이고 연속일 때,

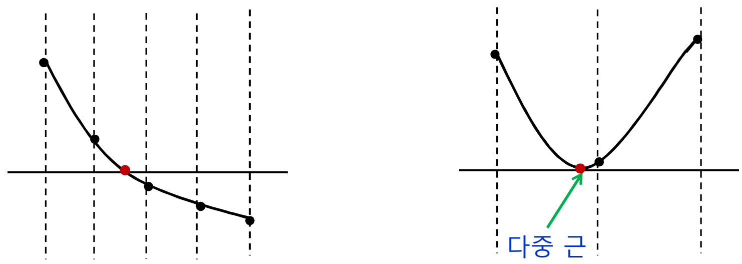
  $f(x_l)$ 과 $f(x_u)$ 가 반대부호 이면 (즉, $f(x_l)f(x_u) < 0$)

  → x_l 와 x_u 사이에 하나 이상의 실근 존재

☞ x_l 와 x_u 사이를 소구간으로 나누어 함수 값 부호가 변하는지 조사

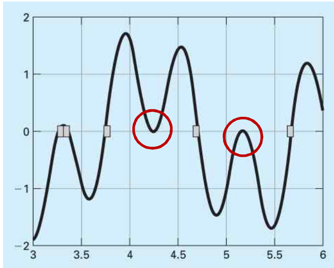
문제점 : 소구간 길이가 너무 짧으면 시간낭비,

〃 너무 길면 가까이 있는 근을 놓칠 수 있음

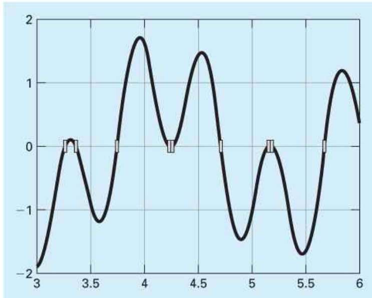



```
function xb = incsearch( func, xmin, xmax, ns)
  x = linspace(xmin, xmax, ns);    % Incremental search
  f = func(x);
  nb =0;   xb =[];
 for k = 1 : length(x)-1
  if sign(f(k)) ~= sign(f(k+1))    % 함수 부호 변화 조사
    nb = nb + 1;
    xb(nb,1) = x(k);
    xb(nb,2) = x(k+1);
  end
 end
 if isempty(xb) % 구간을 발견 못 함
    disp('no brackets found')
 else               % 발견된 구간의 수 출력
    disp('number of brackets:'),  disp(nb)
 end
 end
```

```
>> incsearch(@(x) sin(10*x)+cos(3*x), 3, 6, 50)
  number of brackets:     5
  ans =   3.2449    3.2857
          3.3265    3.3673
          3.7347    3.7755
          4.6735    4.7143
          5.6327    5.6939
```



```
>> 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
```



① Bisection : 구간을 반으로 나누어 함수 값 비교하는 방법, 느리나 확실

Ex) f(m) = `sqrt(9.81*m/0.25)*tanh(sqrt(9.81*0.25/m)*4)-36.`

ε_s =0.5%

- **x_l = 50 , x_u = 200 :** $x_r = \dfrac{50+200}{2} = 125$ → $f(50)f(125) = 1.871$

  x_l ~ x_r 근 없다

  ☞ x_r을 x_l 로 지정

- **x_l = 125 , x_u = 200 :**

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

  → $f(125)f(162.5) = -0.147$

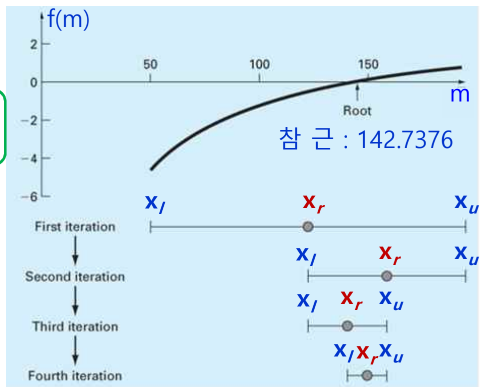
  x_l ~ x_r 근 있다

  ☞ x_r을 x_u 로 지정

  $$\varepsilon_a = \left|\frac{162.5-125}{162.5}\right| \times 100\% = 23.08\%$$

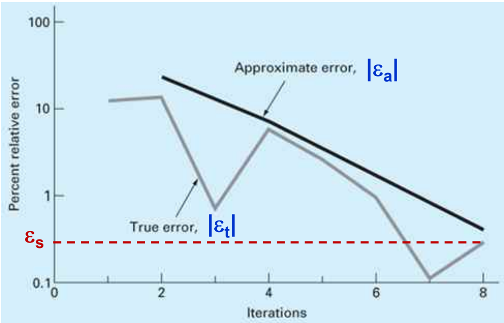
- **x_l = 125 , x_u = 162.5 :**

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



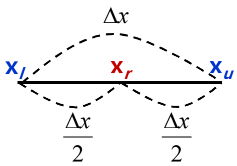
☞ 필요한 정확도 만족까지 계속 반복

Iteration	x_l	x_u	x_r	$\lvert\varepsilon_a\rvert$ %	$\lvert\varepsilon_t\rvert$ %
2	125	200	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



- $\lvert\varepsilon_t\rvert$ 선이 들죽날죽(ragged)함
- $\lvert\varepsilon_a\rvert > \lvert\varepsilon_t\rvert$ 이므로
  $\lvert\varepsilon_a\rvert < \varepsilon_s$ 로 종료되었을 때 ,
  원하는 수준보다 덜 정확

Absolute error 로 오차해석



Bisection 에서 approximate root 는 $x_r = \dfrac{(x_l + x_u)}{2}$

참 근은 $\Delta x = x_u - x_l$ 안에 존재 → $x_r \pm \dfrac{\Delta x}{2}$

Absolute error

- Bisection 시작 전 absolute error : $E_a^0 = x_u^0 - x_l^0 = \Delta x^0$
- 1번 실행 한 후 absolute error : $E_a^1 = \dfrac{\Delta x^0}{2}$
- n번 실행 한 후 absolute error : $E_a^n = \dfrac{\Delta x^0}{2^n}$ → $2^n = \dfrac{\Delta x^0}{E_a^n}$ → $n = \log_2(\dfrac{\Delta x^0}{E_a^n})$

☞ 원하는(desired) 절대 오차($E_{a,d}$)를 달성하기 위한 반복횟수 결정가능

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

예) x_l = 100 , x_u = 200 일 때, $\Delta x^0 = 100$ 이므로

$E_{a,d} = 5$ 라면 $n = \log_2(\dfrac{100}{5}) = \dfrac{\log_{10} 20}{\log_{10} 2} = 4.3219$

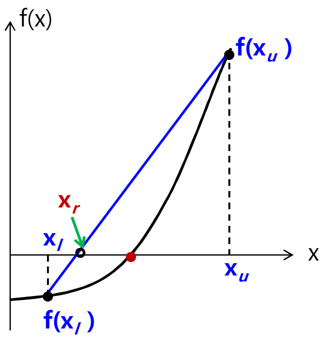
→ 5번 실행

```
function [ root, fx, ea, iter ] = bisect(func, xl, xu,
  es, maxit, varargin)                  % Bisection method
 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;         % 함수 부호가 다르다
  elseif test > 0 ,   xl = xr;     % 함수 부호가 같다
  else      ea = 0;
  end
  if ea <= es | iter >= maxit, break, end
end
 root = xr;     fx = func(xr, varargin{:});
end
```


② **False Position :** 근을 쉽게 찾기 위해 경계의 함수크기를 고려

☞ $(x_l, f(x_l))$ 과 $(x_u, f(x_u))$를 직선으로 연결하여 근의 가짜 위치를 정해 나감

bisection 보다 효과적, 오차가 빨리 감소



- 닮은꼴 삼각형에 의해

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

$\rightarrow$ $(x_r - x_l) f(x_u) = f(x_l)(x_r - x_u)$

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

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

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

☞ $f(x_r)$ 의 부호를 조사하여 x_l 이나 x_u 로 재지정하며 허용오차 이내까지 계속 반복

Ex) Use false position to solve

f(m) = `sqrt(9.81*m/0.25)*tanh(sqrt(9.81*0.25/m)*4)-36.`

- **x_l = 50 , x_u = 200 :**

$$\left[\begin{array}{l} f(x_l) = -4.579387 \\ f(x_u) = 0.860291 \end{array}\right] \rightarrow x_r = 200 - \frac{0.860291(50-200)}{-4.579387 - 0.860291} = 176.2773$$

$f(x_l)f(x_r) = -2.592732$ ← $x_l \sim x_r$ 근 있다 ☞ x_r 을 x_u 로 지정

- **x_l = 50 , x_u = 176.2773 :**

$$\left[\begin{array}{l} f(x_l) = -4.579387 \\ f(x_u) = 0.566174 \end{array}\right] \rightarrow x_r = 176.2773 - \frac{0.566174(50-176.2773)}{-4.579387 - 0.566174} = 162.3828$$

$f(x_l)f(x_r) = -1.5267$ ← $x_l \sim x_r$ 근 있다 ☞ x_r 을 x_u 로 지정

$$\varepsilon_a = \left|\frac{162.3828 - 176.2773}{162.3828}\right| \times 100\% = 8.56\%$$

☞ 허용오차 내까지 계속 반복

False position 이 Bisection 보다 못한 경우

Ex) Locate the root of $f(x) = x^{10} - 1$, x = 0~1.3

- **Bisection**

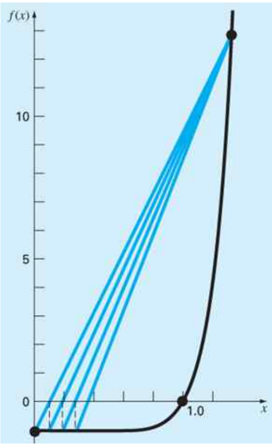
Iteration	x_r	$\lvert\varepsilon_a\rvert$ %	$\lvert\varepsilon_t\rvert$ %
1	0.65	100	35
2	0.975	33.3	2.5
3	1.1375	14.3	13.8
4	1.05625	7.7	5.6

fast

- **False position**

Iteration	x_r	$\lvert\varepsilon_a\rvert$ %	$\lvert\varepsilon_t\rvert$ %
1	0.09430		90.6
2	0.18176	48.1	81.8
3	0.26287	30.9	73.7
4	0.33811	22.3	66.2

slow



☞ 심한 곡률을 가진 함수는 느린 수렴