Lecture Note-Numerical Analysis (3): Roots of Equations-Bracketing Method

1. General Problem Statement of root finding

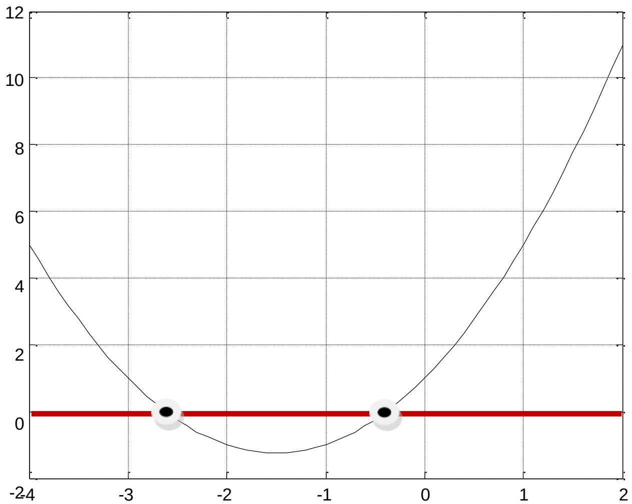
- Problem statement

  Find a vector **x** satisfying the nonlinear equation such as

  $$\mathbf{f}(\mathbf{x}) = 0, \quad \mathbf{f} \in R^n, \quad \mathbf{x} \in R^n, \quad \mathbf{x}_L \le \mathbf{x} \le \mathbf{x}_U$$

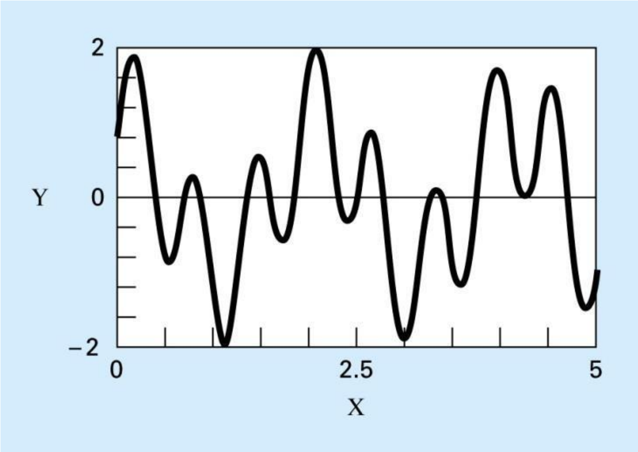
(Example 1) Polynomial equation: $f(x) = x^2 + 3x + 1 = 0, \quad f \in R, \quad x \in R$

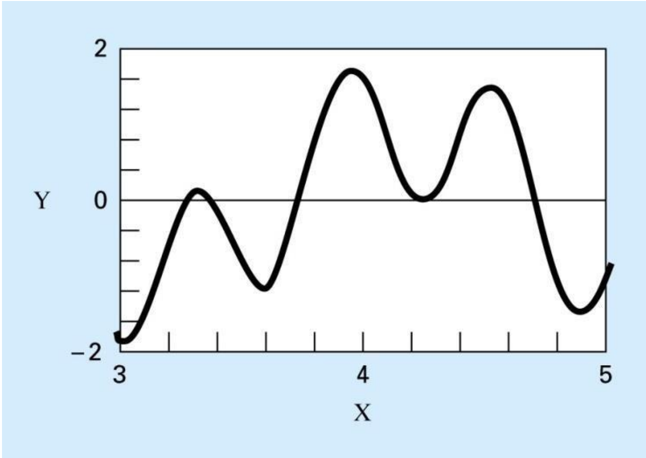
$$\rightarrow \text{Answer} = \quad x = \frac{-3 \pm \sqrt{5}}{2}$$



There are two roots, which can be easily calculated using a standard formula for roots

(Example 2) Multiple roots





(QUESTION 1) How to calculate one root?

(QUESTION 2) How to calculate the roots?

(QUESTION 2) How to calculate the roots with the same value (tangent curve to y=0 line)?

(Example 3) Nonlinear algebraic equation

Problem:

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{pmatrix} = \begin{pmatrix} x_1^2 + x_2^2 - 2 \\ x_1 - x_2 \end{pmatrix} = \mathbf{0}, \quad \mathbf{f} \in R^2, \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in R^2$$

Solution using the substitution method

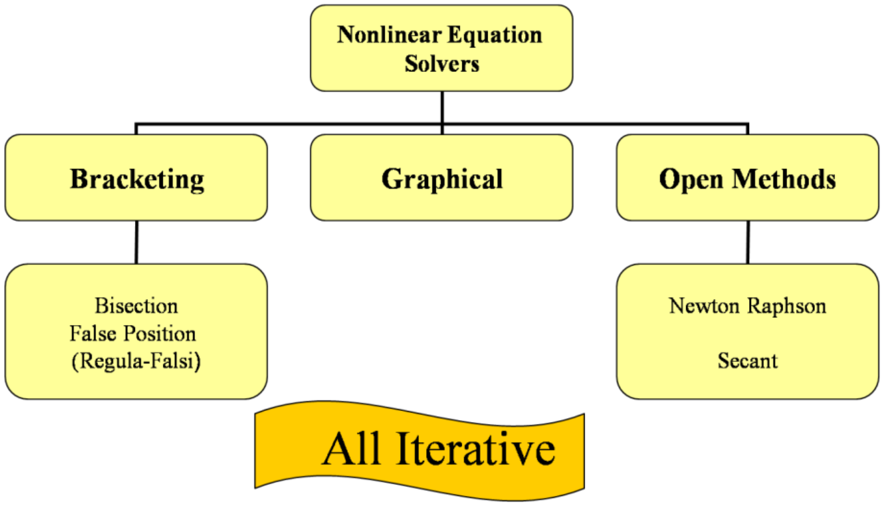
$$x_1 = x_2$$

$$x_1^2 + x_2^2 = 2 \rightarrow 2x_1^2 = 2 \rightarrow x_1^2 = 1 \rightarrow \begin{pmatrix} x_1 = \pm 1 \\ x_2 = \pm 1 \end{pmatrix}$$

Therefore, we have two solutions

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \end{pmatrix}$$

○ Category of the numerical methods for the nonlinear algebraic equation (NAE)



2. Pseudo code to calculate Quadratic equation

- Find real or complex numbers satisfying the quadratic equation $f(x) = ax^2 + bx + c = 0, \quad x \in R, \quad f(x) \in R$

- Solution

$$x = \begin{cases} \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a} & , \quad \text{if } a \neq 0 \\ -\dfrac{c}{b} & , \quad \text{if } a = 0 \text{ and } b \neq 0 \\ \text{no solution} & , \quad \text{if } a = b = 0 \text{ and } c \neq 0 \\ \text{infinite number of solutions} & , \quad \text{if } a = b = c = 0 \end{cases}$$

- Pseudo code

Function Quadroot(a,b,c,r1,r2,i1,i2,nr)

!input: a, b, c→ coefficient of quadratic function

!output: r1,r2,i1,i2,nr→ two complex number and number of roots (if nr=-2, roots are complex pair)

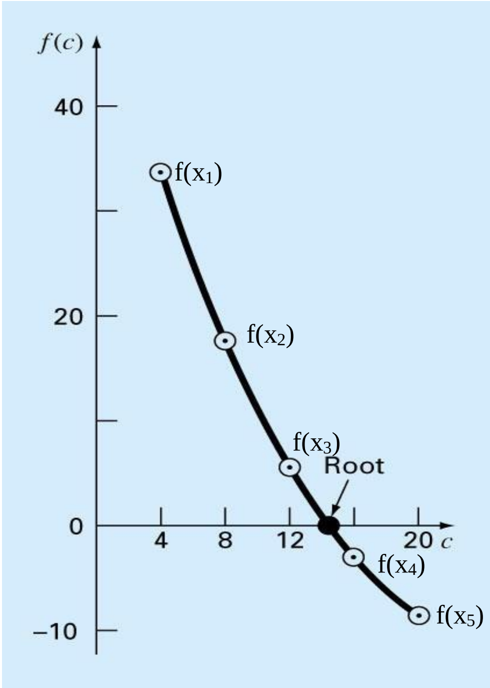
```
If a=0,
    if b=0,
        if c=0, all x can be a solution  ot      (nroot=inf)
        herwise, no x can be a solution          (nroot=0)
    else
        x = -c / b                               (nroot=1)
    end if
else
    d = b^2 - 4ac
    if d < 0,                                    (nroot=-2, negative sign to denote two complex numbers)
        real1= -b/2a          ; imag1  =  sqrt(-d)/2a;
        real2= r1             ;imag2   = - imag1/2a;
    else                                         (nroot= 2, negative sign to denote two complex numbers)
        real1= ( -b+ sqrt(d))/2a    ; imag1  =  0;
        real2= ( -b - sqrt(d))/2a   ;imag2   =  0;
    end if
end if
```

End Quadroot

3. **General Characteristics of a function value around roots: where are roots located?**

When one root is located between $x \in (x_3, x_4)$

$$f(x_1) \times f(x_2) > 0$$
$$f(x_2) \times f(x_3) > 0$$
$$f(x_3) \times f(x_4) \le 0$$
$$f(x_4) \times f(x_5) > 0$$



Multiplication $f(x_3) \times f(x_4)$ **becomes negative or zero, which says that the function values around these points have change in their signs. In this case, one root must be located over** $x \in [x_3, x_4]$ **as a minimum. (Note: Be careful to include the equality c ondition in** $f(x_3) \times f(x_4) \le 0$ **for completeness)**

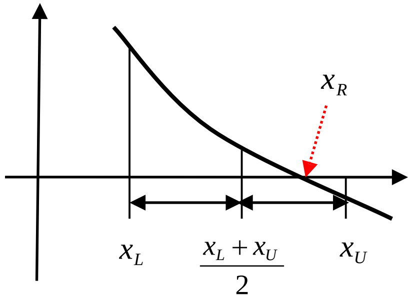
○ Bracketing method needs two points satisfying the condition $f(x_3)\times f(x_4)\le 0$

(QUESTION 1) Where is the root?

$$x_R \in \left(x_L = x_3, \frac{x_3+x_4}{2} \right] \quad \text{or} \quad x_R \in \left(\frac{x_3+x_4}{2}, x_4 = x_U \right)$$

Where x_L is the lower limit of x and x_U is the upper limit of x

→ **Bisection Method : Interval halving method- What interval (left or right) has the root?**



If $f(x_L)f\left(\frac{x_L+x_U}{2}\right)\le 0$ → Left interval

Otherwise → Right interval

4. Bisection method whenever $f(x_L) \times f(x_U) \le 0$ is met

<u>Numerical algorithm</u>

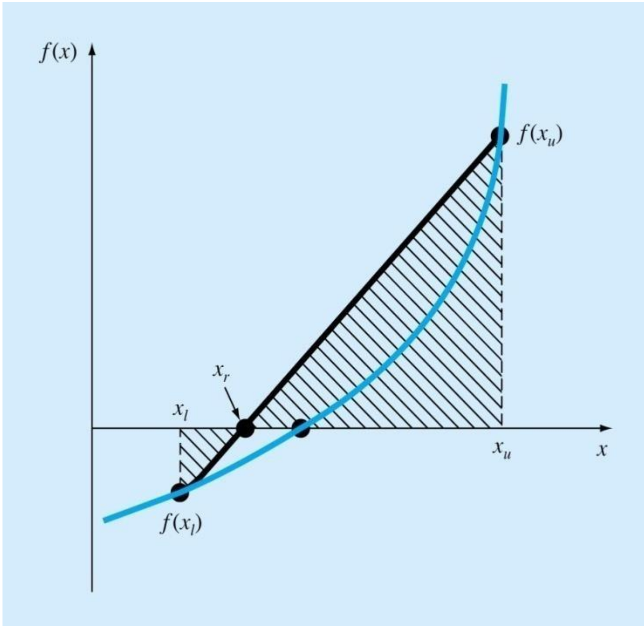
Successively halving the current interval [x_L, x_U] by approximating the roots with 0.5*(x_L+x_U)

```
Function BISECT(xL,   xU, ITmax, xR, epsilon)
! Pseudo code for Bisection method when  f(xL) × f(xU) ≤ 0
! Given variables: xL (Lower x-limit), xU (Upper x-limit), ITmax (maximum iteration)
! Given tolerance: epsilon <<1
! Given external function: f(x)
        fL=f(xL)
        fU= f(xU)
        Do j=1,2,3, …….,ITmax
            xR = 0.5*(xL+xU)                ! estimation of root
            fR = f(xR)                      ! function value at xR
            fT = fL*fR                      ! test function
            if f T >0                       ! no root in [xL, xR]
                xL= xR
                fL= fR
             else                           !one root exists
                xU= xR
                fU= fR
            end if
```

```
! convergence check
          xD = abs(xU- xL)          ! distance between xL and xU
          if xD < epsilon, exit
      End do
      xR = 0.5*(xL+xU)                    ! final result
End BISECT
```

5. Linear interpolation method (False-position method) whenever $f(x_L)\times f(x_U)\le 0$

(QUESTION 2) Is there any more efficient approximation of the root when x_L **and** x_U **are given?**



A linear approximation function between x_L **and** x_U **using the data pairs** $(x_L, f_L),(x_U, f_U)$ **.**

$$f(x)\approx f_L+\left(\frac{f_U-f_L}{x_U-x_L}\right)(x-x_L)=0 \rightarrow x_R\approx x_L-\left(\frac{x_U-x_L}{f_U-f_L}\right)f_L$$

→ **Linear interpolation method (False-position method)**

Numerical algorithm

A linear approximation function between x_L **and** x_U **using the data pairs** (x_L, f_L), (x_U, f_U).

$$f(x) \approx f_L + \left(\frac{f_U - f_L}{x_U - x_L}\right)(x - x_L) = 0 \rightarrow x_R \approx x_L - \left(\frac{x_U - x_L}{f_U - f_L}\right) f_L$$

Function FalsePos(x_L, x_U, ITmax, x_R, epsilon) ! Linear interpolation method

!Pseudo code for False-Position method when $f(x_L) \times f(x_U) \le 0$

! using the **False-position formula** $x_R \approx x_L - \left(\frac{x_U - x_L}{f_U - f_L}\right) f_L$

! Given variables: x_L (Lower x-limit), x_U (Upper x-limit), ITmax (maximum iteration)

! Given tolerance: epsilon <<1

! Given external function: f(x)

!

f_L=f(x_L)

f_U= f(x_U)

Do j=1,2,3, …….,ITmax

!

! only difference in the estimation of root x_R

! x_R = 0.5*(x_L+x_U) at bisection method

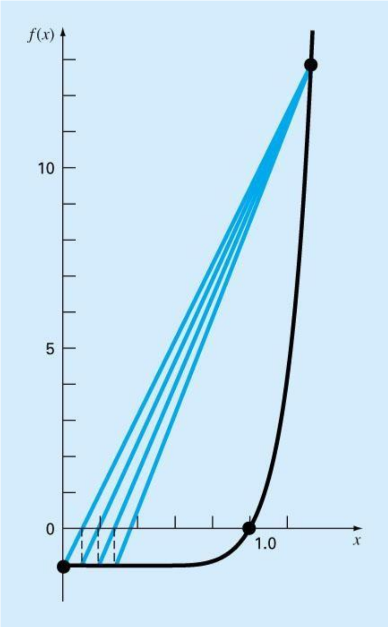
!

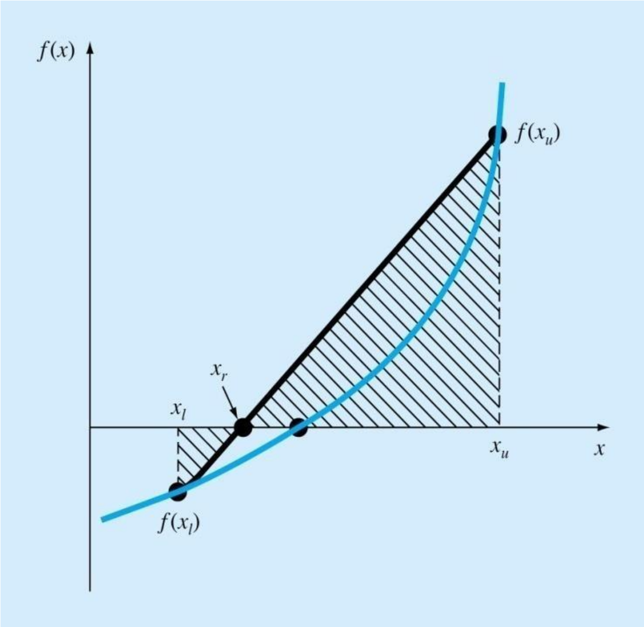
<u>x_R =x_L-(x_U-x_L)*f_L/(f_U-f_L) ! estimation of root using False-position formula</u>

```
      f_R =  f(x_R)                   ! function value at x_R
      f_T =  f_L*f_U                  ! test function
      if f_T > 0                      ! no root in [x_L, x_R]
          x_L= x_R
          f_L= f_R
        else
          x_U= x_R
          f_U= f_R
       end if
! convergence check
      x_D = abs(x_U- x_L)             ! distance between x_L and x_U
      if x_D < epsilon, exit          ! if distance is small enough→ a converged solution
   End do
   x_R = x_L-(x_U-x_L)*f_L/(f_U-f_L)  ! final result
End BISECT
```

Pitfalls of the False-Position Method: slow convergence

$f(x) = x^{10} - 1$ **when** $f'(x \approx 0) = 10x^9 \approx 0$ **is too small**





In this case, the Bisection method is preferable to the False Position method.

6. Convergence of the bisection method

From various experiences, I recommend to use the bisection method rather than the False Position method (linear interpola tion method) because of its robustness in that it guarantees the convergence in any cases. Let's consider the convergence when the initial uncertainty interval is $\Delta x = x_U - x_L$. Then the following reduction in the uncertainty interval can be expected with the bisection method.

1st interval halving: $(\Delta x)_1 = \frac{1}{2}\Delta x$

2nd interval halving: $(\Delta x)_2 = \left(\frac{1}{2}\right)^2 \Delta x$

$\vdots$

k-th interval halving: $(\Delta x)_k = \left(\frac{1}{2}\right)^k \Delta x$

If we want to reduce the uncertainty interval less than $\varepsilon\Delta x$ with the extremely small ε, then we can estimate the number of iteration t o meet the corresponding accuracy requirement as

$$(\Delta x)_k = \left(\frac{1}{2}\right)^k \Delta x \le \varepsilon \Delta x \rightarrow \left(\frac{1}{2}\right)^k \le \varepsilon$$

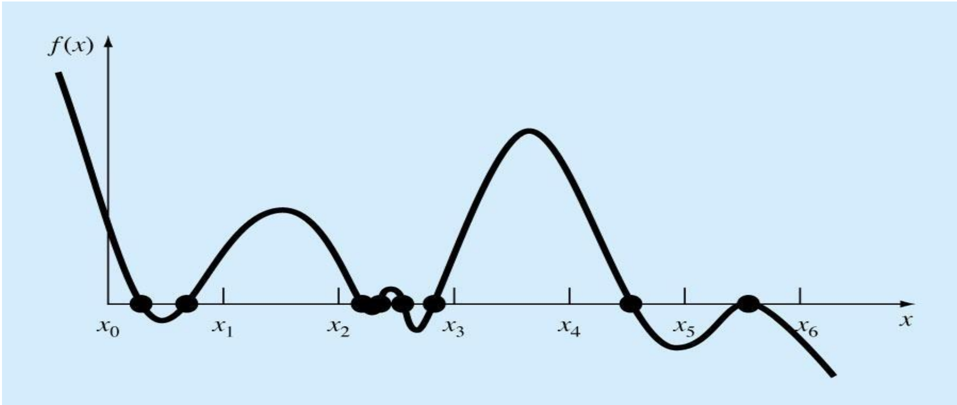
$$\rightarrow -k \log 2 \le \log \varepsilon \quad \rightarrow$$

$$\rightarrow k \ge -\frac{\log \varepsilon}{\log 2}$$

ε	k
10^{-4}	14
10^{-5}	17
10^{-6}	20
10^{-7}	24

Therefore, the bisection method represents exponential convergence in the solution.

7. Incremental search to find several different roots



- If we want to apply previous root finding algorithms, we should find an interval which meets the existence condition of one root such as $f(x_L) \times f(x_U) \le 0$

Numerical algorithm to find several roots in $x \in [x_{\min}, x_{\max}]$

1. Divide the total interval into N element such as

$$\Delta x = (x_{\max} - x_{\min})/(N-1)$$
$$x_j = x_{\min} + (j-1)\Delta x, \quad j = 1,2,\cdots,N$$
$$I_j = [x_j, x_{j+1}]$$

2. Determine whether each interval has one root by checking the condition, $f(x_j) \times f(x_{j+1}) \le 0$.
3. If the condition is met, perform the root finding,
   Otherwise skip the interval $I_j = [x_j, x_{j+1}]$ by assuming no roots exist in this interval

Pitfalls of above method

1. Fine intervals (nodes, grid) should be used to find the most of roots
2. We can’t guarantee whether all root was found or not, even with highly fine node system (See Figure)

Function IncremSearch(x_{MIN}, x_{MAX}, N, IT_{max}, x_R, N_{root}, epsilon)

1. **Set** $\Delta x = (x_{\max} - x_{\min})/(N-1),\ j = 1,\ x_L = x_1,\ f_L = f(x_1)$, **$N_{root}$=0**

2. **Repeat j=2,3,4,5,…., N**

   $x_U = x_L + \Delta x$

   $f_L = f_U$

   $f_U = f(x_U)$

   If f_L*$f_U \le$ 0, then

   N_{root} = N_{root} + 1 **! number of roots**

   Call BISECT(x_L, x_U, IT_{max}, x_{Root}, epsilon) **! finding one root**

   x_R(N_{root}) = x_{Root} **! (N_{root})-th root**

   end if

End IncremSearch