**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



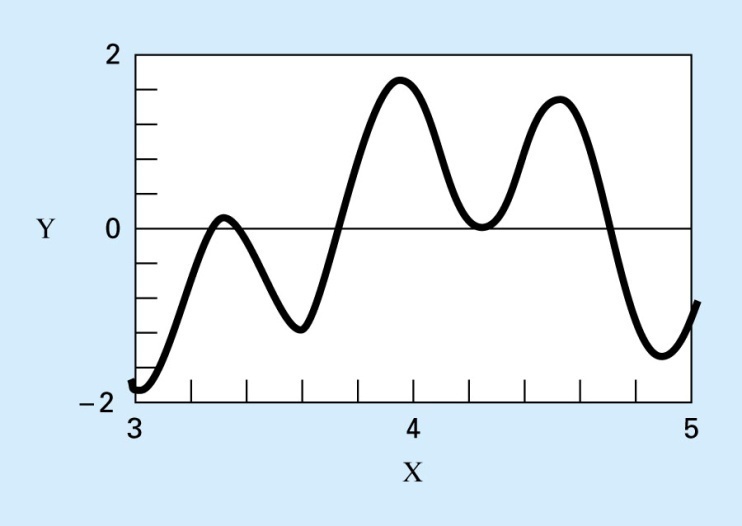
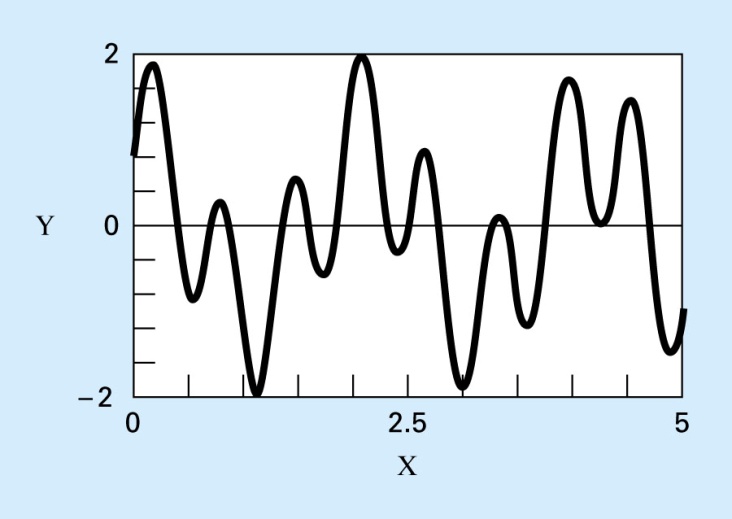
(Example 1) Polynomial equation: 

🡪 Answer = 



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:



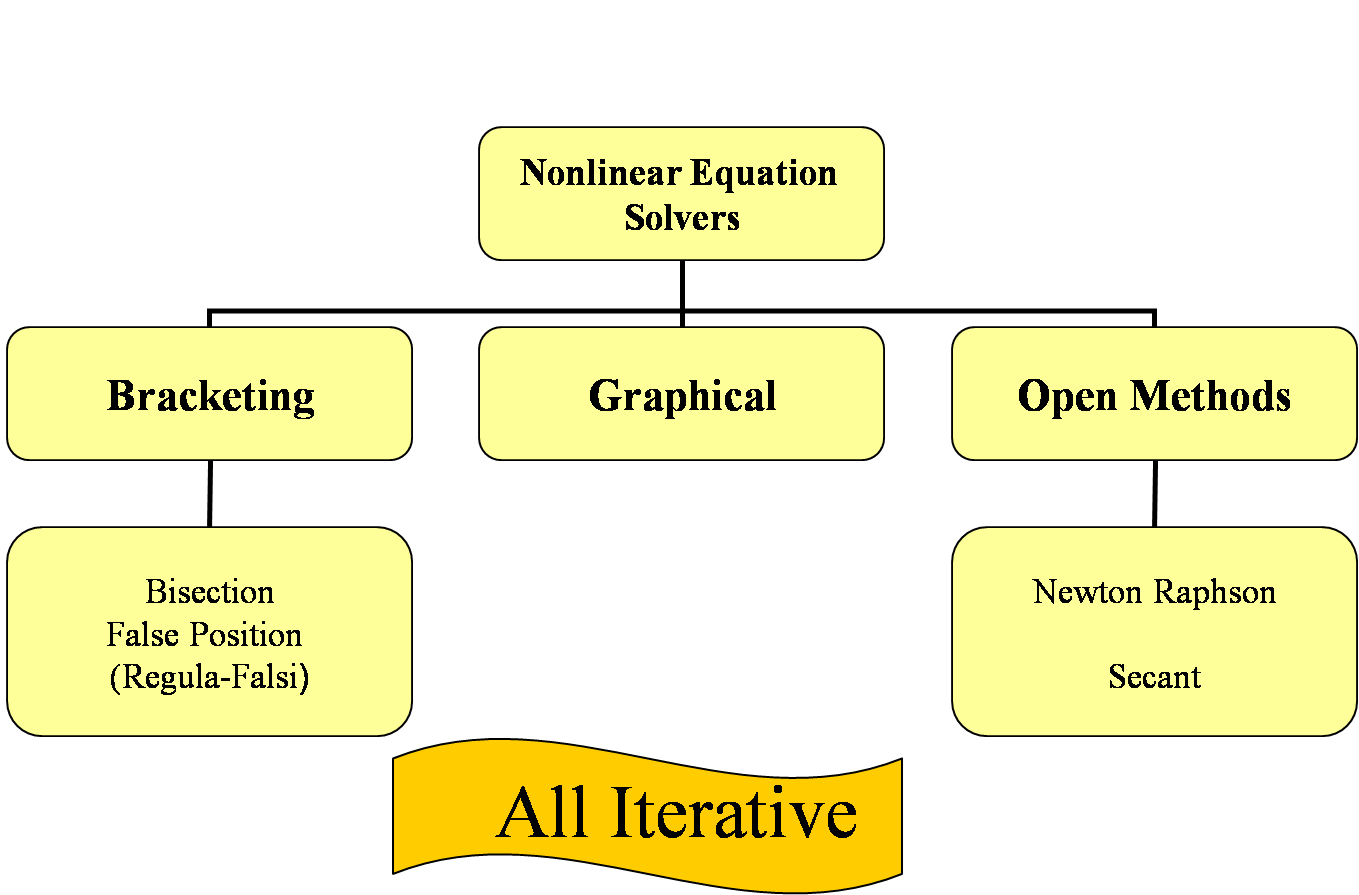
Solution using the substitution method



Therefore, we have two solutions



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



1. **Pseudo code to calculate Quadratic equation**

* Find real or complex numbers satisfying the quadratic equation 
* Solution



* 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 (nroot=inf)

otherwise, no x can be a solution (nroot=0)

else

 (nroot=1)

end if

else



if , (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;

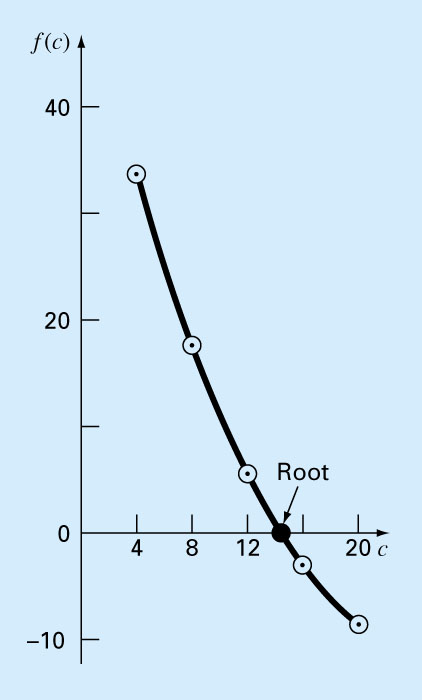
end if

end if

**End Quadroot**

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

**When one root is located between **



f(x1)

f(x2)

f(x3)

f(x4)



f(x5)

**Multiplication 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  as a minimum. (Note: Be careful to include the equality condition in****for completeness)**

* Bracketing method needs two points satisfying the condition ****

**(QUESTION 1) Where is the root?**

 or 

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

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











1. **Bisection method whenever is met**

**Numerical algorithm**

**Successively halving the current interval [xL, xU] by approximating the roots with 0.5\*(xL+xU)**

**Function BISECT(xL, xU, ITmax, xR, epsilon)**

**! Pseudo code for Bisection method when** 

**! 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**

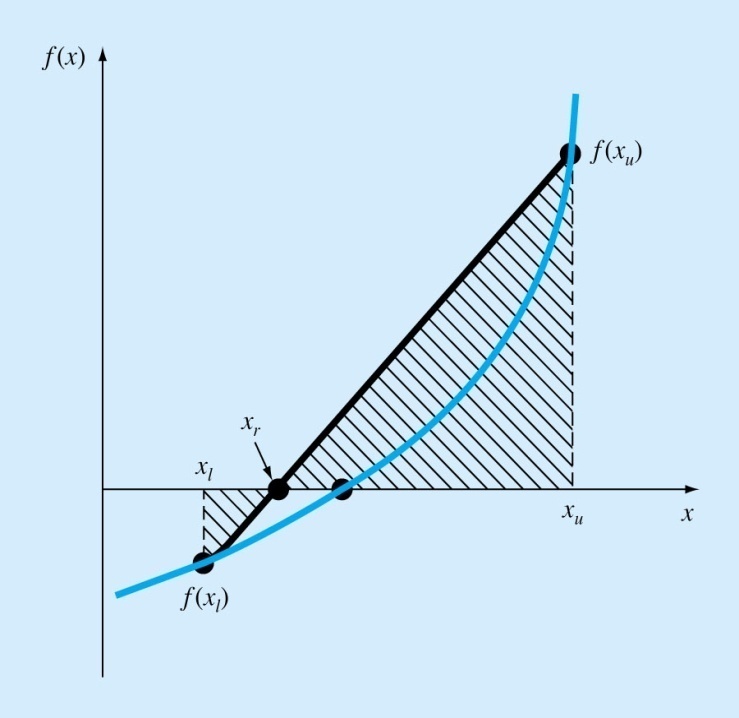
**End do**

**xR = 0.5\*(xL+xU) ! final result**

**End BISECT**

1. **Linear interpolation method (False-position method) whenever** 

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

****

**A linear approximation function between and  using the data pairs,.**

**🡪**

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

**Numerical algorithm**

**A linear approximation function between and  using the data pairs, .**

**🡪**

**Function FalsePos(xL, xU, ITmax, xR, epsilon) ! Linear interpolation method**

**!Pseudo code for False-Position method when** 

**!** using the **False-position formula **

**! 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**

**!**

**! only difference in the estimation of root xR**

**! xR = 0.5\*(xL+xU) at bisection method**

**!**

**xR =xL-(xU-xL)\*fL/(fU-fL) ! estimation of root using False-position formula**

**fR = f(xR) ! function value at xR**

**fT = fL\*fU ! test function**

**if fT** > **0 ! no root in [xL, xR]**

**xL= xR**

**fL= fR**

**else**

**xU= xR**

**fU= fR**

**end if**

**! convergence check**

**xD = abs(xU- xL) ! distance between xL and xU**

**if xD < epsilon, exit ! if distance is small enough🡪 a converged solution**

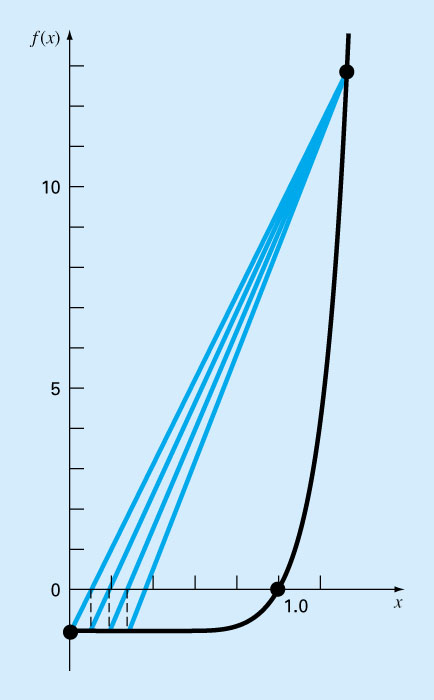
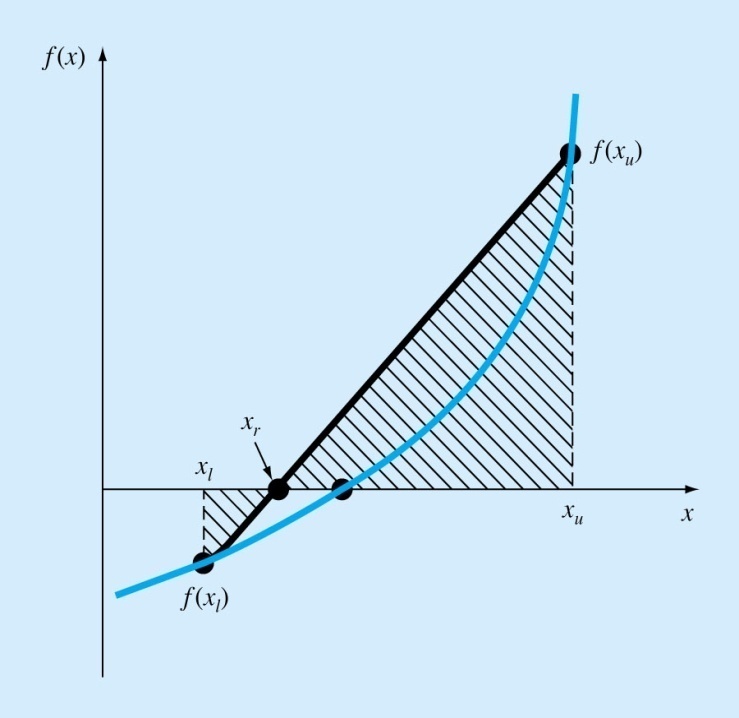
**End do**

**xR = xL-(xU-xL)\*fL/(fU-fL) ! final result**

**End BISECT**

**Pitfalls of the False-Position Method: slow convergence**

** when is too small**

** **

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

1. **Convergence of the bisection method**

From various experiences, I recommend to use the bisection method rather than the False Position method (linear interpolation 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****. Then the following reduction in the uncertainty interval can be expected with the bisection method.

1st interval halving: ****

2nd interval halving: ****

****

k-th interval halving: ****

If we want to reduce the uncertainty interval less than with the extremely small , then we can estimate the number of iteration to meet the corresponding accuracy requirement as

** 🡪**

10-4 14

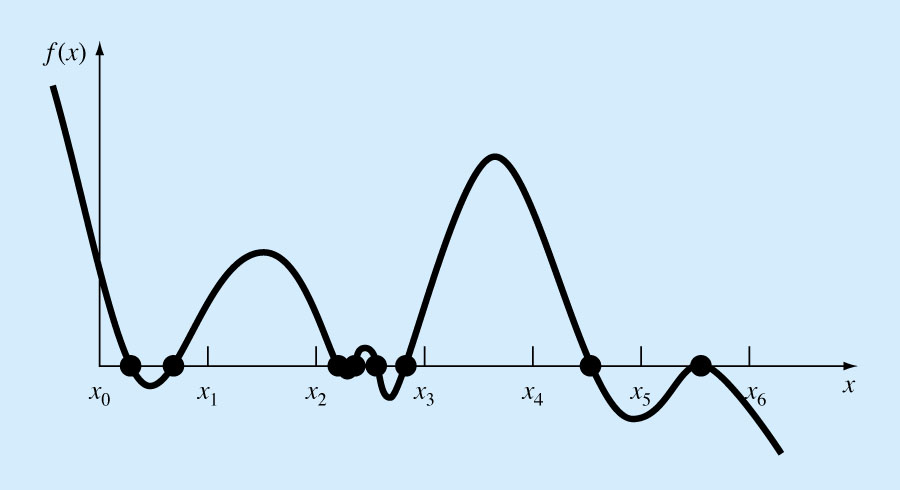
10-5 17

10-6 20

10-7 24

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

1. **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 

**Numerical algorithm to find several roots in **

1. Divide the total interval into N element such as



1. Determine whether each interval has one root by checking the condition, .
2. If the condition is met, perform the root finding,

Otherwise skip the interval  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(xMIN, xMAX, N, ITmax, xR, Nroot, epsilon)**

1. **Set , Nroot=0**
2. **Repeat j=2,3,4,5,…., N**

****

**If fL\*fU**  **0, then**

**Nroot = Nroot + 1 ! number of roots**

**Call BISECT(xL, xU, ITmax, xRoot, epsilon) ! finding one root**

**xR(Nroot) = xRoot ! (Nroot)-th root**

**end if**

**End IncremSearch**