MATH 304-Numerical Analysis and Optimization ---Nonlinear Equation Solver

Instructor: Peng Sun, Ph.D.

Email: peng.sun568@dukekunshan.edu.cn

Reference book

- S. C. Chapra, R. P. Canale, Numerical methods for Engineers (seventh edition), McGraw-Hill Education, 2015, Chapter 5,6

Outline

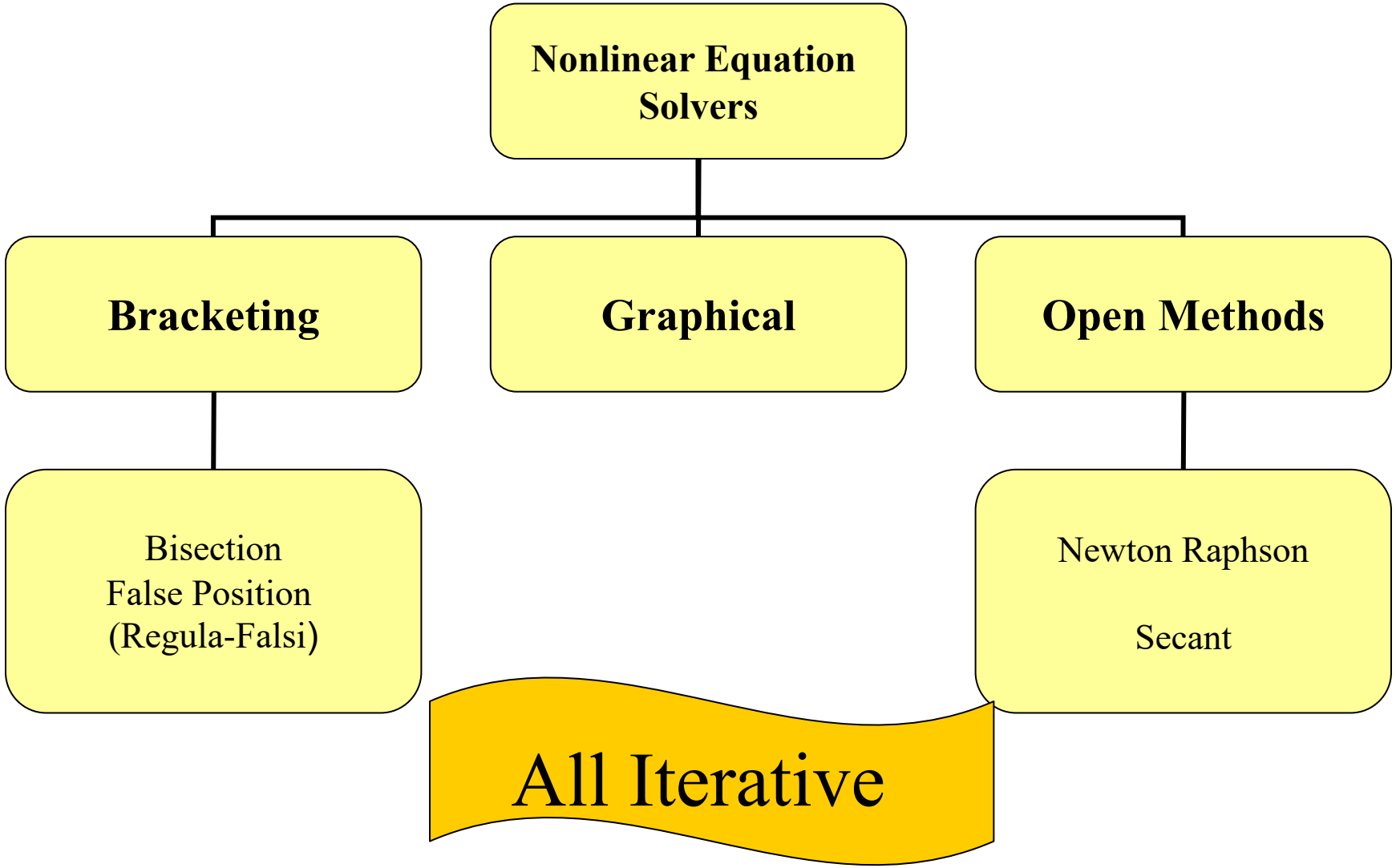
- Introduction
- Bracketing method
- Open method

Introduction

- Solver of equations f(x)=0
  - In the previous lecture, f(x) is linear, i.e., $f(x) = AX - B$
  - In this lecture, we will extend to the case that f(x) is a ***nonlinear equation***.

- Features of nonlinear equation:
  - **Closed form solution cannot be derived for a general nonlinear equation**.
  - **Iterative algorithm must be applied to find an approximation solution**, i.e., numerical solution.
    - Bracketing solution
    - Open solution

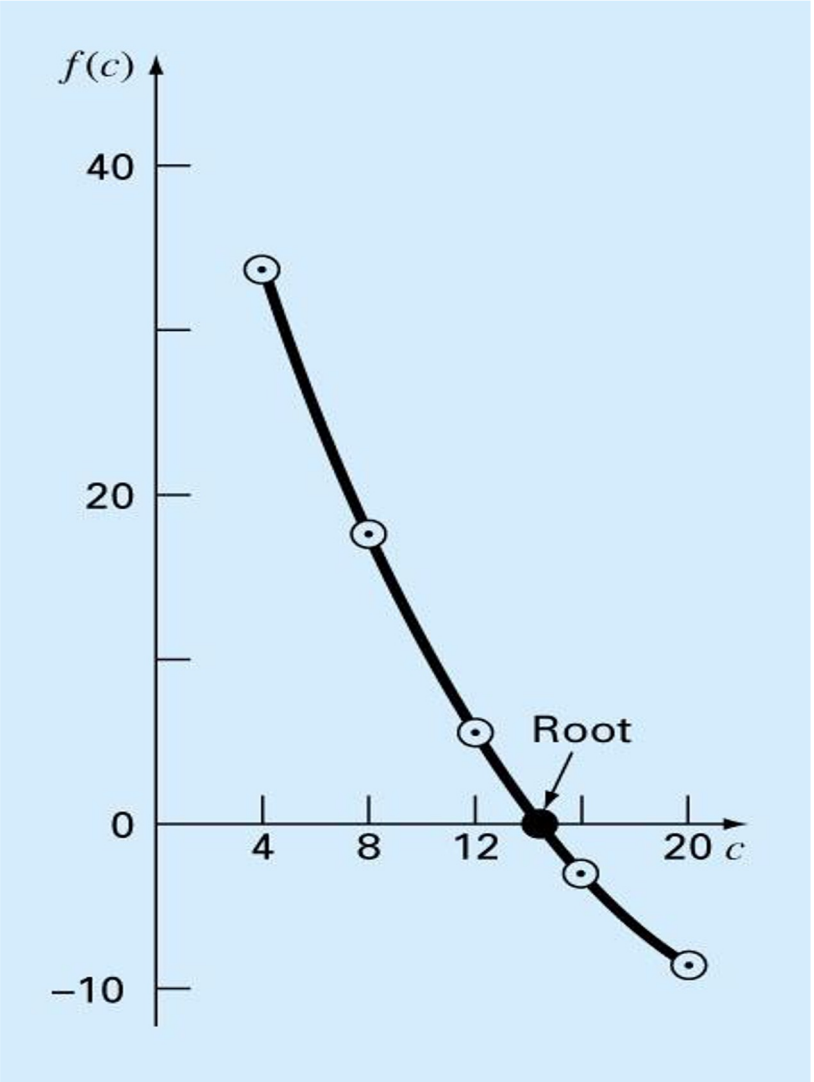
Introduction



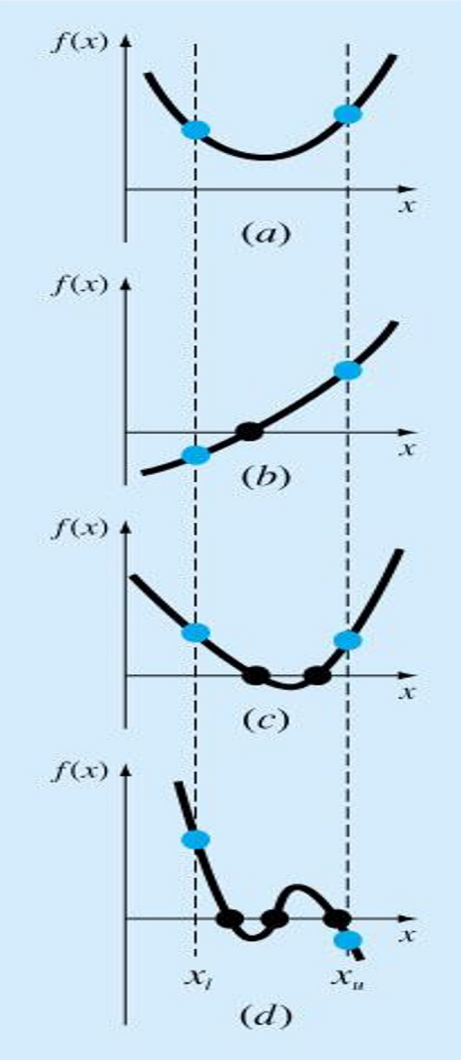
Introduction

Graphical method

- Simple and straightforward method
- Obtaining an estimation of the root of the equation f(x)=0 by plotting the function and observing where it crosses the x axis.
- Graphical techniques are of limited practical value because they are not precise. However, graphical methods can be utilized to ***obtain rough estimates of roots***. These estimates can be employed as starting guesses for numerical methods.



Introduction

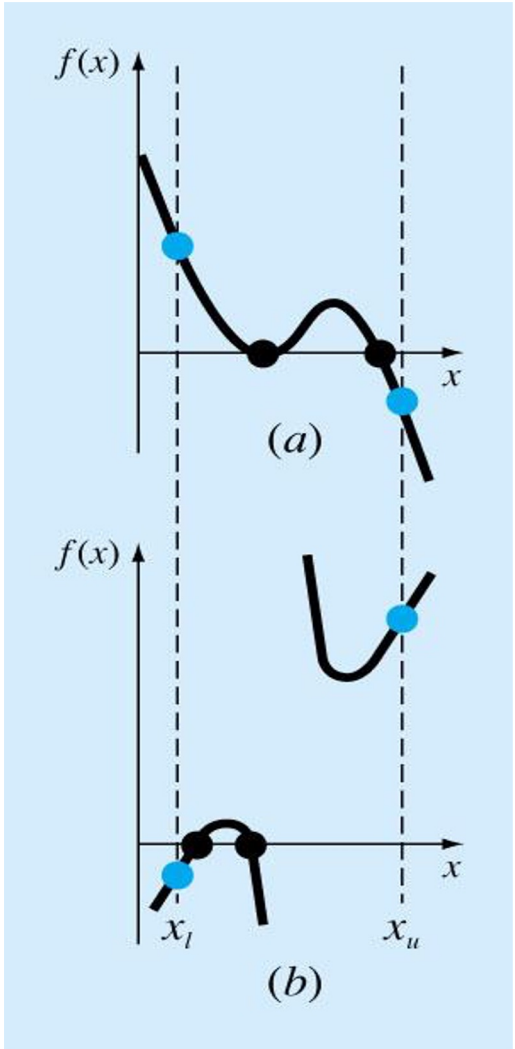


No answer (No root)

Nice case (one root)

Oops!! (two roots!!)

Three roots(Might work for a while!!)



Two roots(Might work for a while!!)

Discontinuous function. Need special method

Bracketing method

- Motivation
  - The bracketing method exploits the fact that a function typically changes sign in the vicinity of a root.

- Key idea

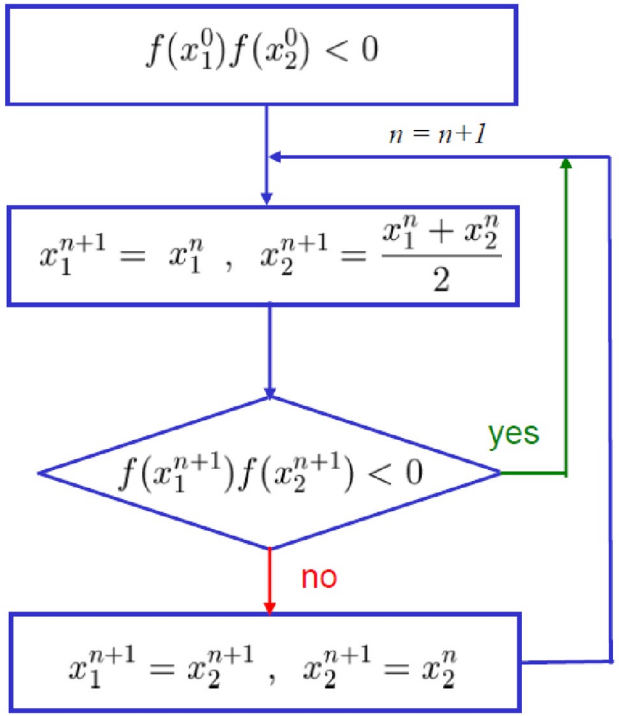
1. Assuming that the solution of f(x)=0 is within an interval [a, b]
2. Iteratively shrink [a, b] to find the solution x

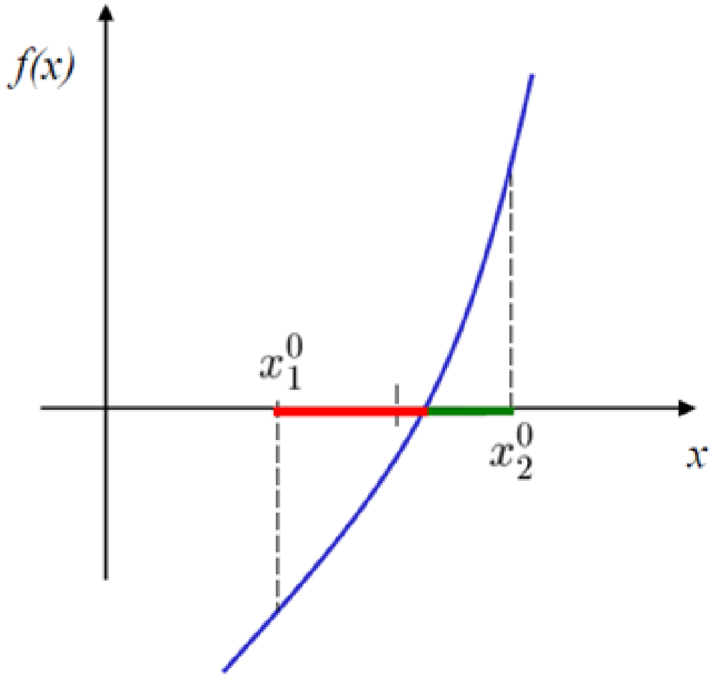
Bracketing method

1. Bisection method
2. False Position Method

Bracketing method: Bisection method

- Basic logic:
  - "Keep dividing in two for the interval within which at least one root lies"



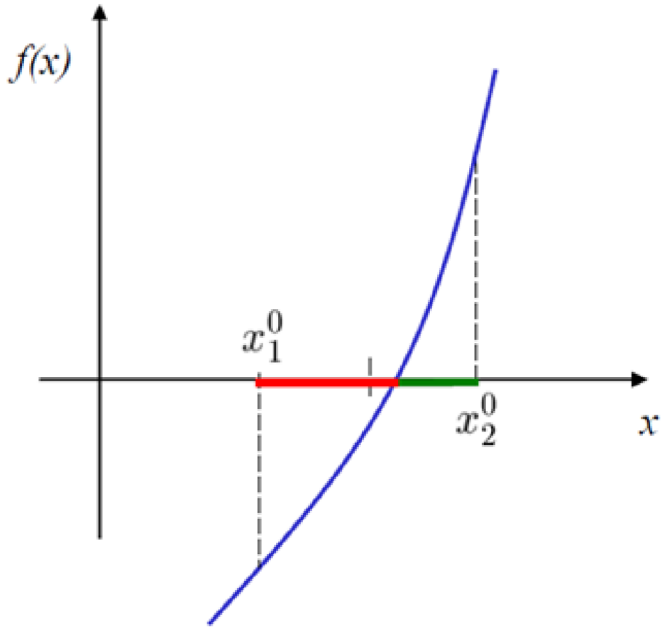


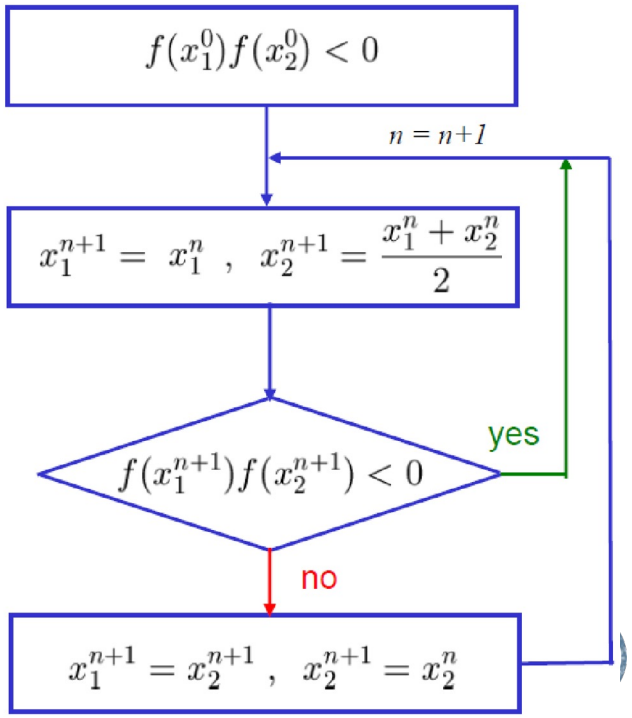
Bracketing method: Bisection method

- Algorithm:

For the arbitrary equation of one variable, f(x)=0

1. Pick x_1 and x_2 such that they bound the root of interest, check if $f(x_1).f(x_2) < 0$.
2. Estimate the root by evaluating $f[(x_1+x_2)/2]$.
3. Find the pair
   - If $f(x_1).\ f[(x_1+x_2)/2]<0$, root lies in the lower interval, then $x_2=(x_1+x_2)/2$ and go to step 2.
   - If $f(x_1).\ f[(x_1+x_2)/2]>0$, root lies in the upper interval, then $x_1= [(x_1+x_2)/2$, go to step 2.
   - If $f(x_1).\ f[(x_1+x_2)/2]=0$, then root is $(x_1+x_2)/2$ and terminate.

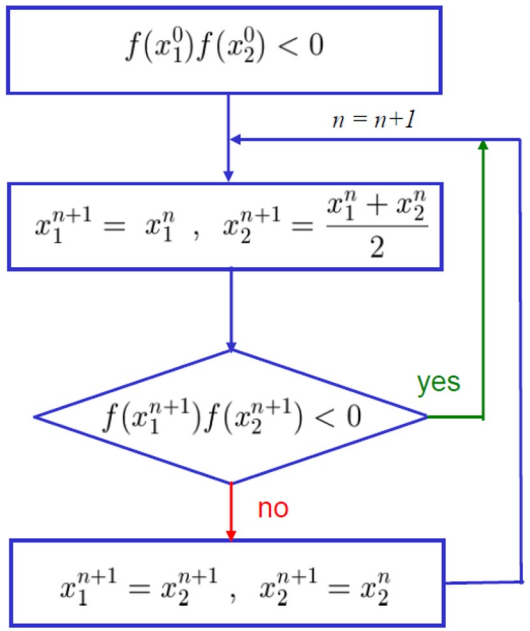


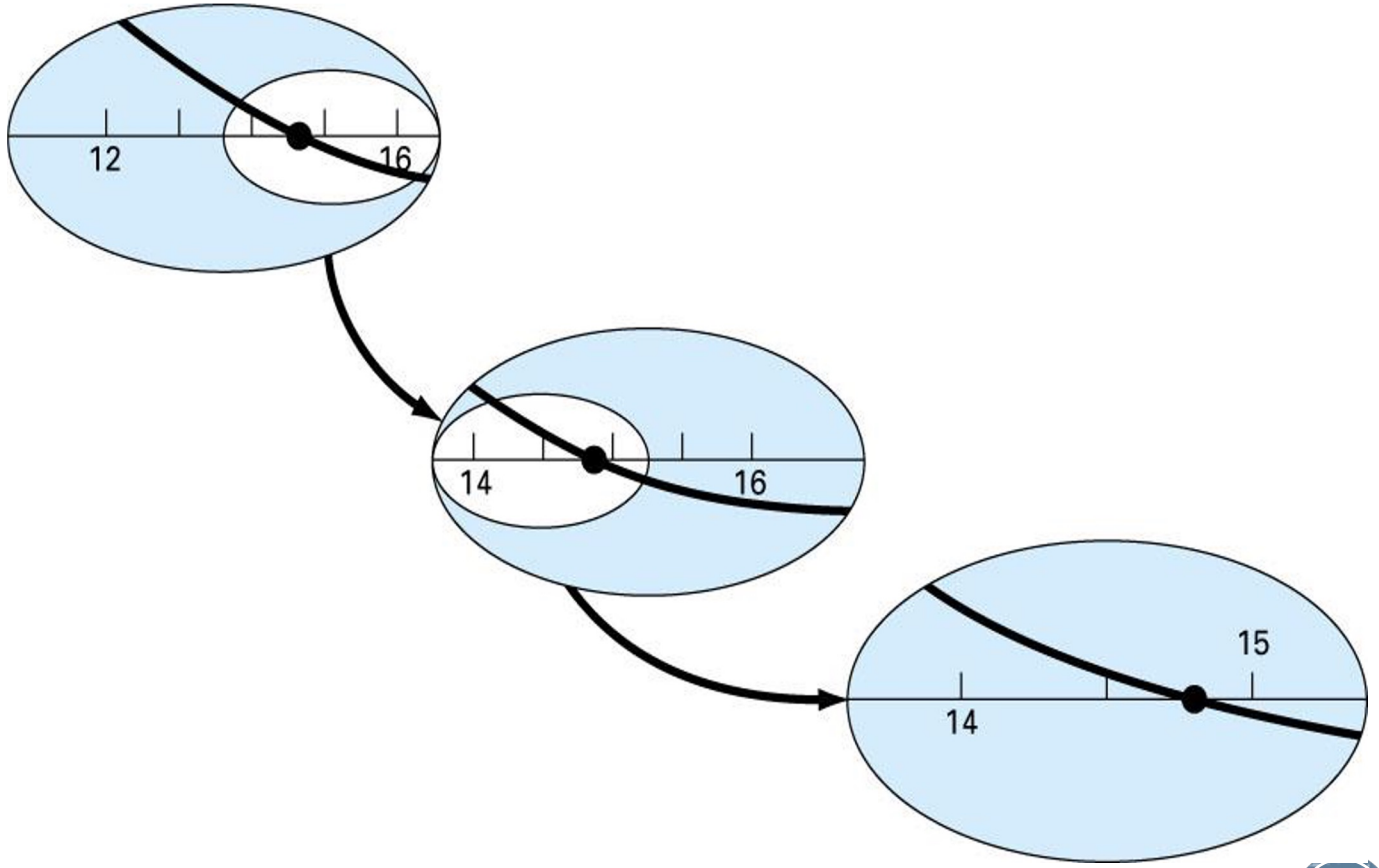


Bracketing method: Bisection method

Example:

- For a nonlinear equation f(x), with the initial guess that solution $x \in [12, 16]$





Bracketing method: Bisection method

- Algorithm (continued)

4. Compare the prespecified percent tolerance ε_s with ε_a
5. If $\varepsilon_a < \varepsilon_s$, stop. Otherwise repeat the process.

ε_a denotes approximate percent relative error, which is given by,

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

$$\Rightarrow \quad \varepsilon_a = \left| \frac{x_1 - \frac{x_1 + x_2}{2}}{\frac{x_1 + x_2}{2}} \right| \times 100\% \quad or \quad \varepsilon_a = \left| \frac{x_2 - \frac{x_1 + x_2}{2}}{\frac{x_1 + x_2}{2}} \right| \times 100\%$$

Bracketing method: Bisection method

- Termination criteria and error estimates

1. Relative error: stop when relative estimated error, i.e., relative added value, is negligible

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

2. Absolute error: absolute added values go to zero: $\Delta x \sim 0$
   - But we do not know the true solution
   - Each successive iteration halves the maximum error
   - Estimate the number of iterations required to attain an absolute error

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

   - Δx^0 is the initial error (e.g., the first interval), $E_{a,d}$ is the desired absolute error.

Bracketing method: Bisection method

- Termination criteria and error estimates
- Example:
- If the desired absolute magnitude of the error is $\frac{\varepsilon_s \times x}{100\%} = 10^{-4}$ and $\Delta x^0 = 2$, how many iterations will you have to do to get the required accuracy in the solution.
- solution:

$$10^{-4} = \frac{2}{2^k} \Rightarrow 2^k = 2 \times 10^4 \Rightarrow \quad k \cong 14.3 = 15$$

Bracketing method: Bisection method

- Evaluation of method

Pros

- Easy
- Always find root
- Number of iterations required to attain an absolute error can be computed a priori.

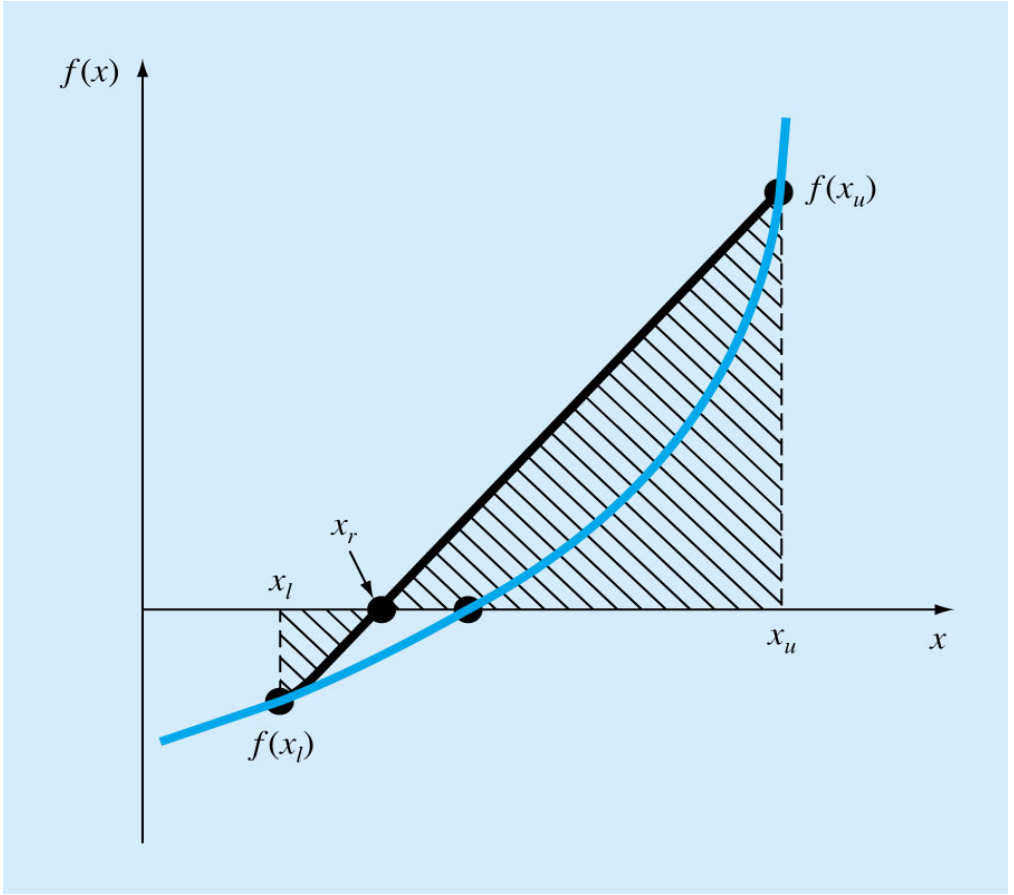
Cons

- Slow
- Know a and b that bound root
- Multiple roots
- No account is taken of $f(x_l)$ and $f(x_u)$, if $f(x_l)$ is closer to zero, it is likely that root is closer to x_l .

Bracketing method: False position method

- Bisection method is a kind of brute force scheme
  - Somewhat inefficient approach
  - Does not account the magnitudes of $f(x_l)$ and $f(x_u)$

- False position method (***linear interpolation method***)
  - Motivation
  - If a real root is bounded by x_l and x_u of f(x)=0, then we can approximate the solution by doing a linear interpolation between the points [x_l, f(x_l)] and [x_u, f(x_u)] to find the x_r value such that l(x_r)=0, l(x) is the linear approximation of f(x).



Bracketing method: False position method

- Algorithm:

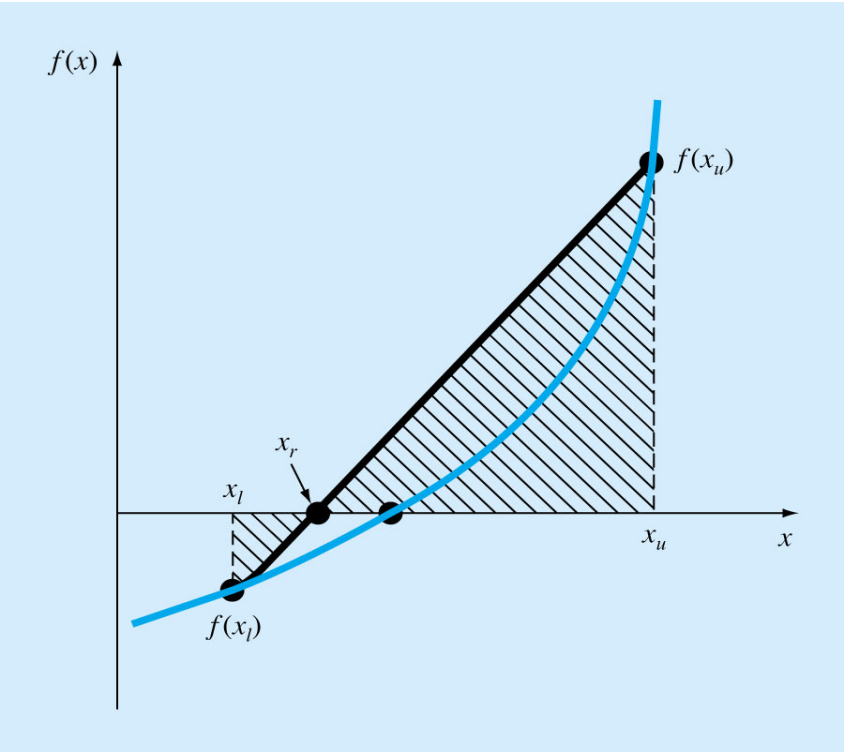
1. Find a pair of values of x, x_l and x_u such that $f_l=f(x_l)$ <0 and $f_u=f(x_u)$ >0.
2. Estimate the value of the root from the following formula,

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

$$\Rightarrow \quad \frac{f(x_L)}{(x_r - x_L)} = \frac{f(x_U)}{(x_r - x_U)}$$

$$\Rightarrow \quad x_r = x_U - \frac{f(x_U)(x_L - x_U)}{f(x_L) - f(x_U)}$$

and evaluate $f(x_r)$.



Bracketing method: False position method

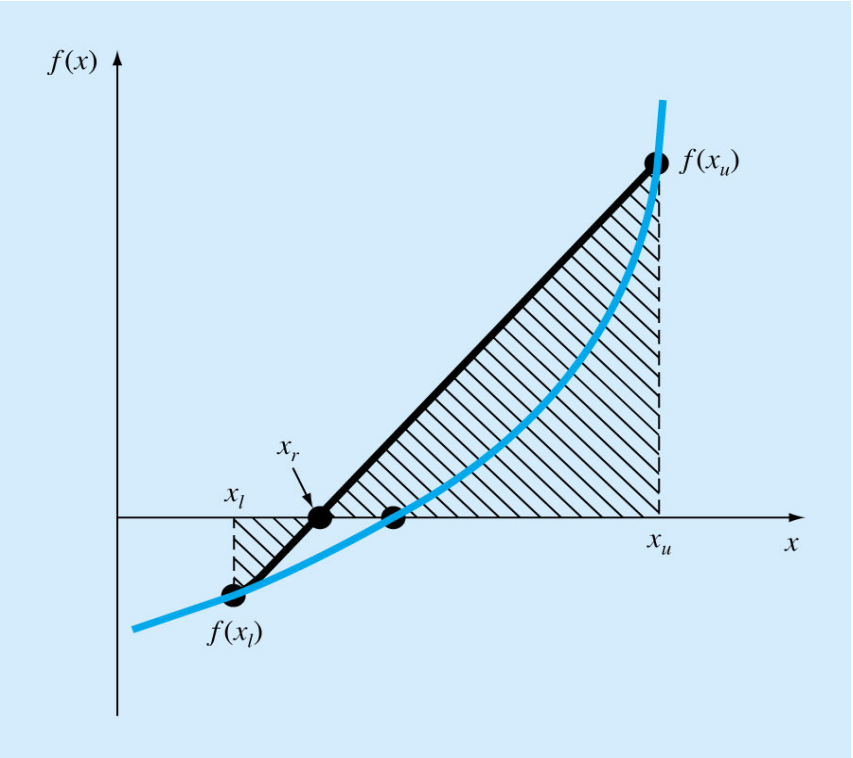
- Algorithm: (continued)

3. Use the new point to replace one of the original points, keeping the two points on opposite sides of the x axis.

   If $f(x_r)<0$ then $x_l=x_r$ ==> $f_l=f(x_r)$

   If $f(x_r)>0$ then $x_u=x_r$ ==> $f_u=f(x_r)$

   If $f(x_r)=0$ then you have found the root and need go no further!



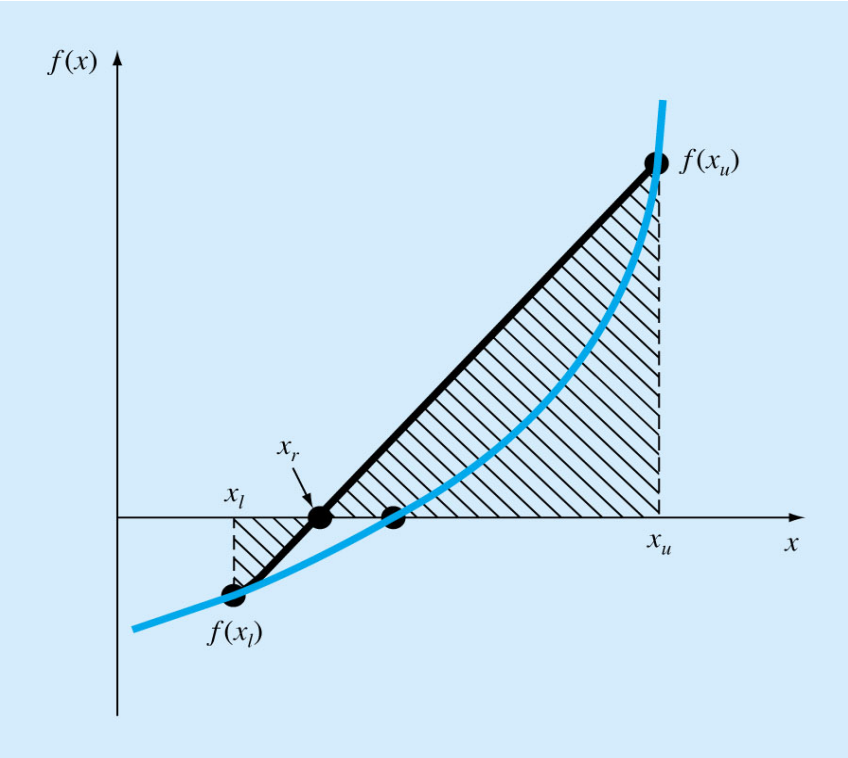
Bracketing method: False position method

- Algorithm: (continued)

4. See if the new x_l and x_u are close enough for convergence to be declared. If they are not go back to step 2.

- Why this method?
  - Faster
  - Always converges for a **single root**.

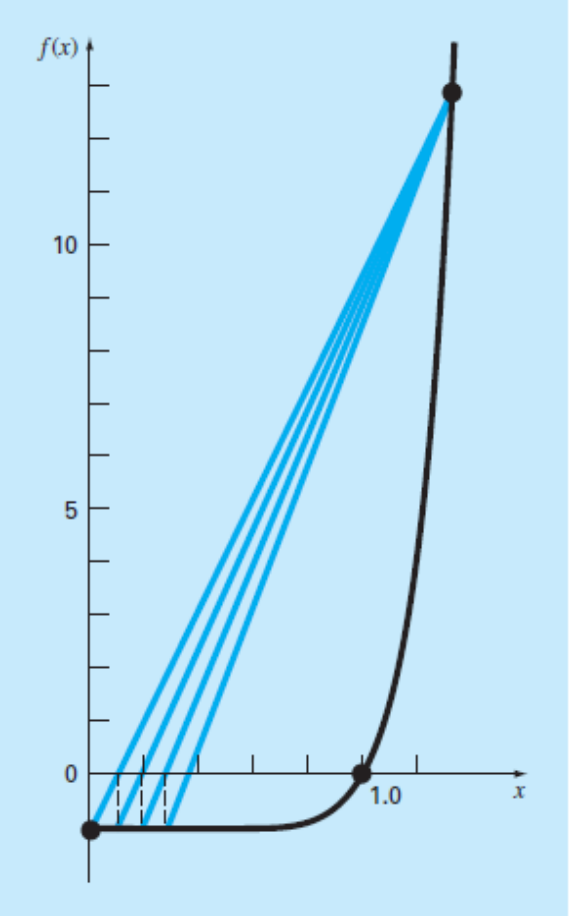
Note: Always check by substituting estimated root in the original equation to determine whether $f(x_r) \approx 0$.



Bracketing method: False position method

Pitfalls of the False-position method

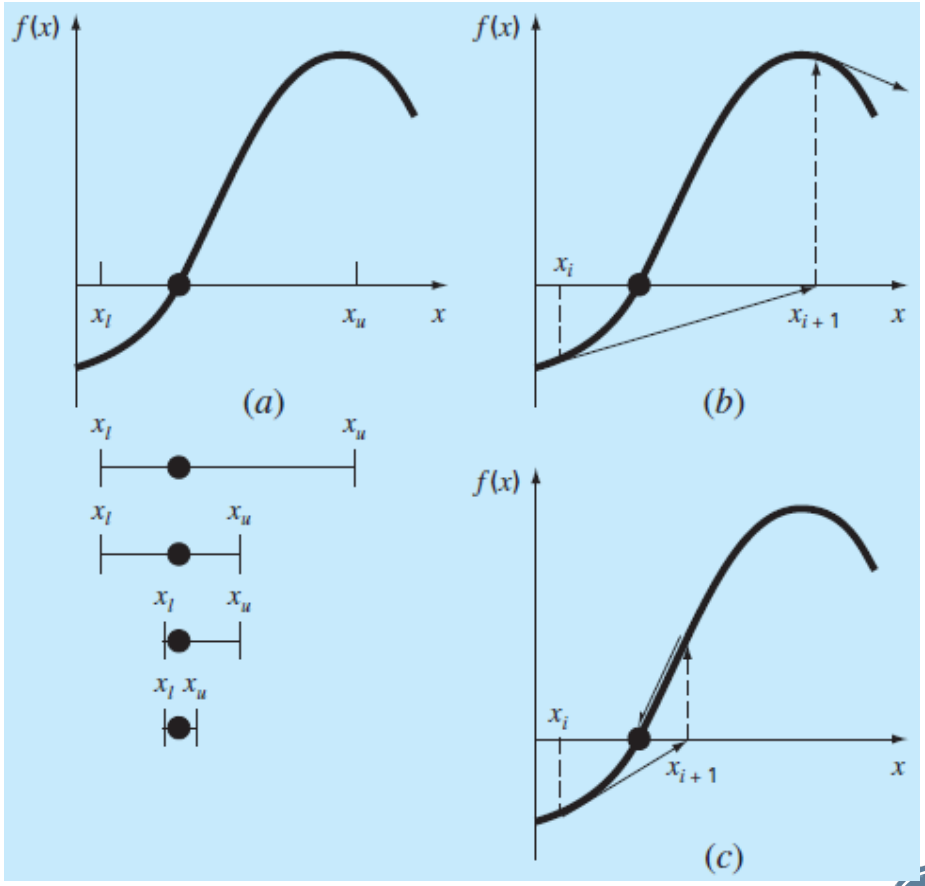
- Error of False position can decrease much faster than that of Bisection, but if the function is far from a straight line, False Position will be slow (one of the end point remains fixed).
  - Example of $f(x) = x^{10} - 1$
- Remedy: Modified False Position
  - If end point remains fixed for a few iterations, reduce end-point function values at end point, e.g., divided by 2



Open method

- Motivation:
  - The bracketing method, by assuming an interval bounded by a lower and an upper bound (necessarily bracket the root), iteratively shrink an interval to find the solution.
    1. Slow to converge;
    2. Without considering the function itself.

- Open method
  - Exploiting formulas that requires only a single start point of x or two starting points that do not necessarily bracket the root.
  - If the method converge, it normally do so much more quickly than the bracketing method.



Open method

1. Fixed-point iteration

2. Newton Raphson method

Open method: Fixed-point iteration

- Rearrange the function so that x is on the left side of the equation:

1. $f(x) = 0 \Rightarrow x = g(x) \Rightarrow g(x) = f(x) + x$
2. $x_k = g(x_{k-1}),\ with\ known\ intial\ x_0$ and $k = 1, 2, \ldots$

- Example:
  - $f(x) = x^2 - x - 2\ and\ x > 0$
  - $\Rightarrow g(x) = x^2 - 2\ or\ g(x) = \sqrt{x+2}\ \ or\ \ g(x) = 1 + \frac{2}{x}$

- Bracketing methods are “convergent”.
- Fixed-point methods may sometime “diverge”, depending on the stating point (initial guess) and how the function behaves.

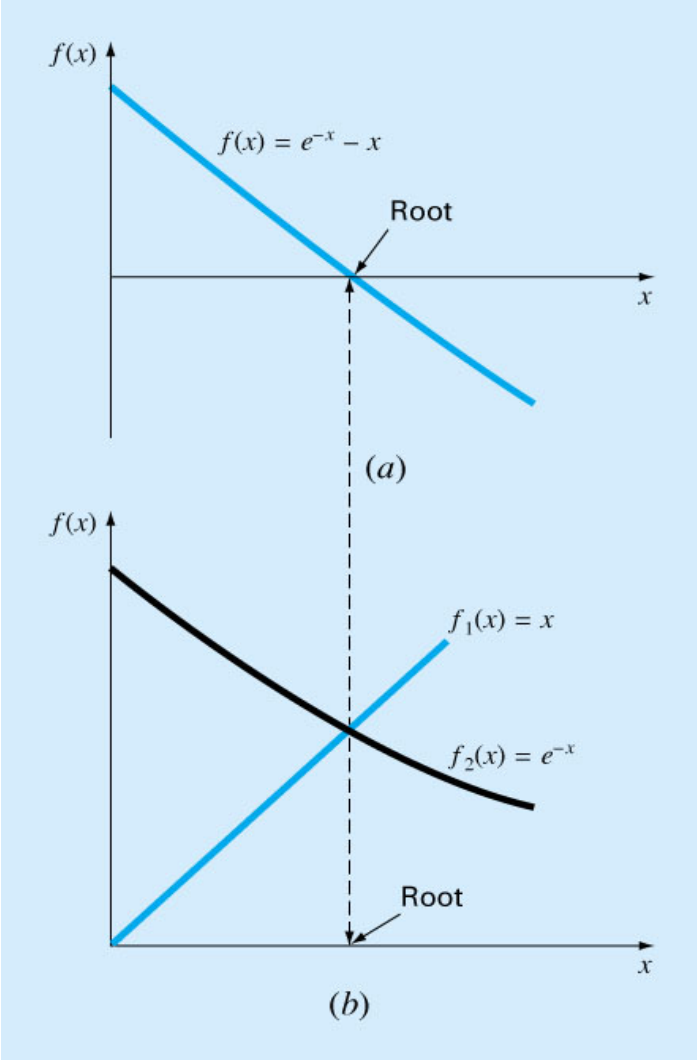
Open method: Fixed-point iteration

Convergence

- x=g(x) can be expressed as a pair of equations:

1. y_1=x
2. y_2=g(x) (component equations)

- Plot them separately.

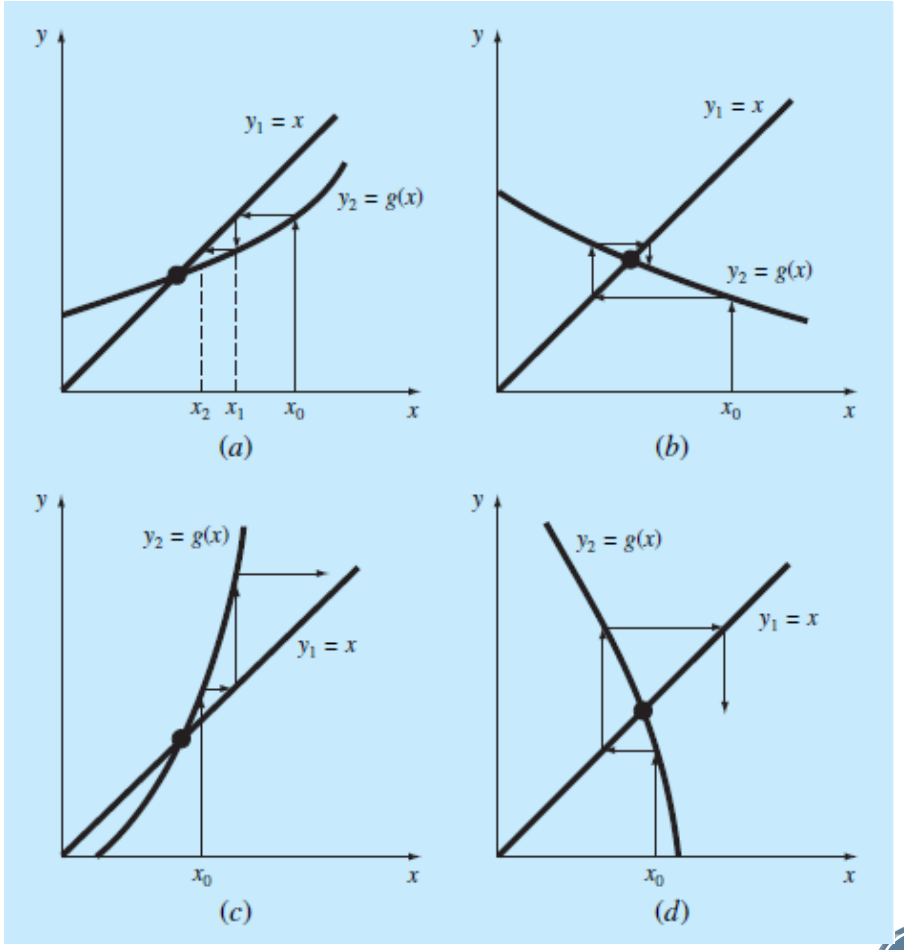


Open method: Fixed-point iteration

Convergence

- Sufficient condition:
  - Fixed-point iteration converges if
$$|g'(x)| < 1$$

- When the method converges, the error is roughly proportional to or less than the error of the previous step, therefore it is called "**linearly convergent.**"



Open method: Fixed-point iteration

Proof of the convergence condition:

- Based on the definition, we have $x_{k+1} = g(x_k)$. Suppose that the true solution is $x_r = g(x_r)$, by taking subtraction, we have

$$x_r - x_{k+1} = g(x_r) - g(x_k) \quad (1)$$

- Relying on the ***derivation mean theorem***, we have

$$g'(\xi) = \frac{g(x_r) - g(x_k)}{x_r - x_k}, where\ \xi \in [x_k, x_r] \quad (2)$$

if a function $g(x)$ and its first derivative are continuous over an interval $a <= x <= b$, then there exists at least one value of x = within the interval satisfies …

- By substituting (1) into (2), we have

$$x_r - x_{k+1} = (x_r - x_k)g'(\xi)$$

- If the true error of iteration k is defined as $E_{t,k} = x_r - x_k$, we have,

$$E_{t,k+1} = E_{t,k}g'(\xi)$$

Consequently, if $|g'(x)| < 1$, the errors decrease with each iteration; for $|g'(x)| > 1$, the error grows.

Open method: Fixed-point iteration

- Aitken acceleration:

Assume that, $|e_n| = |x_n - r| = K^{n-1}e_1$ *or* $x_n = r + K^{n-1}e_1$

Similarly, if $|e_{n+1}| = |x_{n+1} - r| = K^n e_1$ and $|e_{n+2}| = |x_{n+2} - r| = K^{n+1}e_1$, then we have,

$$x_{n+1} = r + K^n e_1 \text{ and } x_{n+2} = r + K^{n+1}e_1$$

Substitute these two equations into

$$\frac{x_n x_{n+2} - x_{n+1}^2}{x_{n+2} - 2x_{n+1} + x_n}$$

We have

$$\frac{x_n x_{n+2} - x_{n+1}^2}{x_{n+2} - 2x_{n+1} + x_n} = \frac{(r + K^{n-1}e_1)(r + K^{n+1}e_1) - (r + K^n e_1)^2}{r + K^{n+1}e_1 - 2(r + K^n e_1) + r + K^{n-1}e_1} = \frac{r(K^{n+1} - 2K^n + K^{n-1})e_1}{(K^{n+1} - 2K^n + K^{n-1})e_1} = \text{r}$$

From three successive estimates of the roots, x_1, x_2, and x_3, we extrapolate to an improved estimate.

However, since the assumption of constant ratio is not normally true, our extrapolated value is not exact. But it is usually improved. We can use this technique by calculating two values begin with x_1, extrapolating, calculating two new values, extrapolating again, etc.

Open method: Fixed-point iteration

- Aitken acceleration (continued):

Define

$$\Delta x_i = x_{i+1} - x_i$$

$$\Delta^2 x_i = \Delta(\Delta x_i) = \Delta(x_{i+1} - x_i) = \Delta x_{i+1} - \Delta x_i = x_{i+2} - 2x_{i+1} + x_i$$

The acceleration scheme becomes

$$r = \frac{x_n x_{n+2} - x_{n+1}^2}{x_{n+2} - 2x_{n+1} + x_n} = x_n - \frac{(\Delta x_n)^2}{\Delta^2 x_n}$$

Open method: Fixed-point iteration

- Aitken acceleration (continued):
- Example: $f(x) = x^2 - 2x - 3 = 0$; we select $g(x) = \sqrt{2x+3}$ and $x_0 = 4$

Fixed-point without acceleration	Fixed-point with Aitken acceleration	Δx	$\Delta^2 x$
$x_0 = 4$	$x_0 = 4$		
$x_1 = \sqrt{11} = 3.316$	$x_1 = \sqrt{11} = 3.316$	0.684	
$x_2 = \sqrt{9.632} = 3.104$	$x_2 = \sqrt{9.632} = 3.104$	0.212	0.472
$x_3 = \sqrt{9.208} = 3.034$	The accelerated estimate is $r = 4.000 - \frac{(0.684\ )^2}{0.472} = 3.009$		
$x_4 = \sqrt{9.068} = 3.011$	We jumped ahead about two iterations.		
$x_5 = \sqrt{9.022} = 3.004$			

Open method: Fixed-point iteration

- Aitken acceleration (continued):
- When to use the Aitken acceleration?

Suppose for some n, we have $x_n, x_{n+1}, x_{n+2}, x_{n+3}$, then evaluate the following,

$$C = \frac{\sum x_i x_{i+1} - \frac{1}{3}\sum x_i \sum x_{i+1}}{\sqrt{\left(\sum x_i^2 - \frac{1}{3}(\sum x_i)^2\right)\left(\sum x_{i+1}^2 - \frac{1}{3}(\sum x_{i+1})^2\right)}}$$

Where the sums are from $i = n$ to $i = n + 1$. If C is close to ± 1, then Aitken acceleration is most effective.

Open method: Fixed-point iteration

- Aitken acceleration (continued):
- Example: $f(x) = x^2 - 2x - 3 = 0$; we select $g(x) = \sqrt{2x+3}$ and $x_1 = 4$

We have,

$$x_0 = 4.000$$
$$x_1 = 3.316$$
$$x_2 = 3.104$$
$$x_3 = 3.034$$

We find that with $n = 0$ in the formula,

$$C = \frac{\sum_{i=0}^{2} x_i x_{i+1} + \frac{1}{3}\sum_{i=0}^{2} x_i \sum_{i=0}^{2} x_{i+1}}{\sqrt{\left(\sum_{i=0}^{2} x_i^2 - \frac{1}{3}\left(\sum_{i=0}^{2} x_i\right)^2\right)\left(\sum_{i=0}^{2} x_{i+1}^2 - \frac{1}{3}\left(\sum_{i=0}^{2} x_{i+1}\right)^2\right)}} \frac{32.974 - \frac{1}{3}\times 10.42 \times 9.454}{\sqrt{(36.631 - 36.1921)(29.8358 - 29.7927)}}$$
$$= 0.99992$$

Open method: Newton-Raphson method

- The Newton-Raphson method is the most widely used root-locating method.
- The Newton-Raphson method is designed based on Taylor series expansion:

$$f(x_{i+1}) = f(x_i) + f'(x_i)\Delta x + f''(x_i)\frac{\Delta x^2}{2!} + O\Delta x^3$$

The root is the value of x_{i+1} when $f(x_{i+1}) = 0$

Rearranging,

Solve for

$$0 = f(x_i) + f'(x_i)(x_{i+1} - x_i)$$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

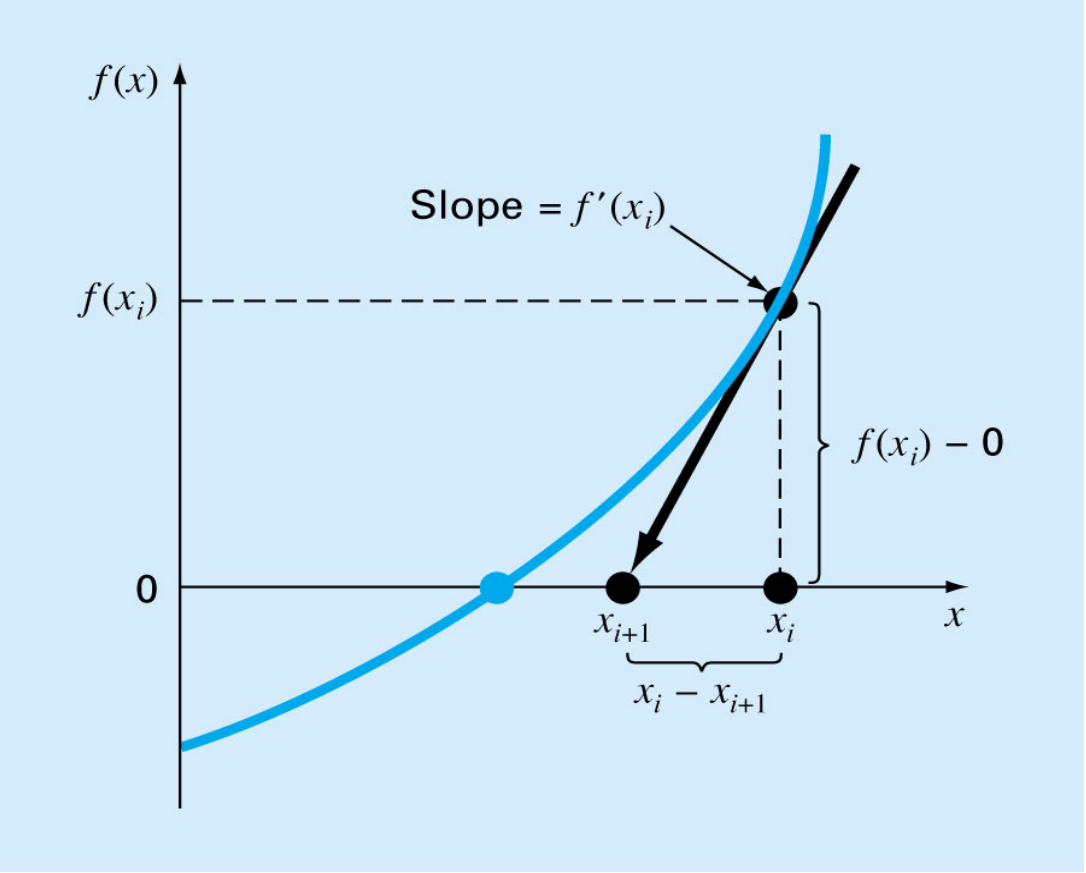
Newton-Raphson formula

Open method: Newton-Raphson method

Geometrical interpretation:

- Using the slope to approximate the original function
- The updating rule:

$$f'(x_i) = \frac{f(x_i) - 0}{x_i - x_{i+1}} \Rightarrow x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$



Open method: Newton-Raphson method

Example: f(x) = ln(x) = 0, with initial guess x0=0.1

Solution:

- Initialization: f'(x) = 1/x, f(x0) = -2.30, and f'(x) = 10;
- Iteration #1:
  1. x1=x0-f(x0)/f'(x0)=0.33
  2. f(x1) = ln(0.33) = -1.11, and f'(x1) = 1/0.33 = 3
- Iteration #2:
  1. x2=x1-f(x1)/f'(x1)=0.70
  2. f(x2) = ln(0.70) = -0.36 and f'(x2)=1/0.70=1.43
- Iteration #3:
  1. x3 = x2-f(x2)/f'(x2)=0.95 → Closing to the true value 1.
  2. f(x2) = ln(0.95) = -0.05 and f'(x2)=1/0.95=1.05

Newton-Raphson formula

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Open method: Newton-Raphson method

Convergence of Newton-Raphson method

- Suppose the true solution is x_r, so $f(x_r) = 0$. Based on the Taylor expansion, we have

$$0 = f(x_i) + f'(x_i)(x_r - x_i) + \frac{f''(\xi)}{2!}(x_r - x_i)^2 \quad (1)$$

- The estimation x_{i+1} satisfies:

$$0 = f(x_i) + f'(x_i)(x_{i+1} - x_i) \quad (2)$$

- By substituting (2) into (1), we have,

$$0 = f'(x_i)(x_r - x_{i+1}) + \frac{f''(\xi)}{2!}(x_r - x_i)^2$$

- If the true error of iteration i is defined as $E_{t,i} = x_r - x_i$, and we assume convergence and $x_i \; and \; \xi$ should be eventually approximated to the root x_r , we have

$$0 = f'(x_i)E_{t,i+1} + \frac{f''(\xi)}{2!}E_{t,i}^2 \Rightarrow E_{t,i+1} = \frac{-f''(x_r)}{2f'(x_r)}E_{t,i}^2$$

Quadratic convergence

Open method: Newton-Raphson method

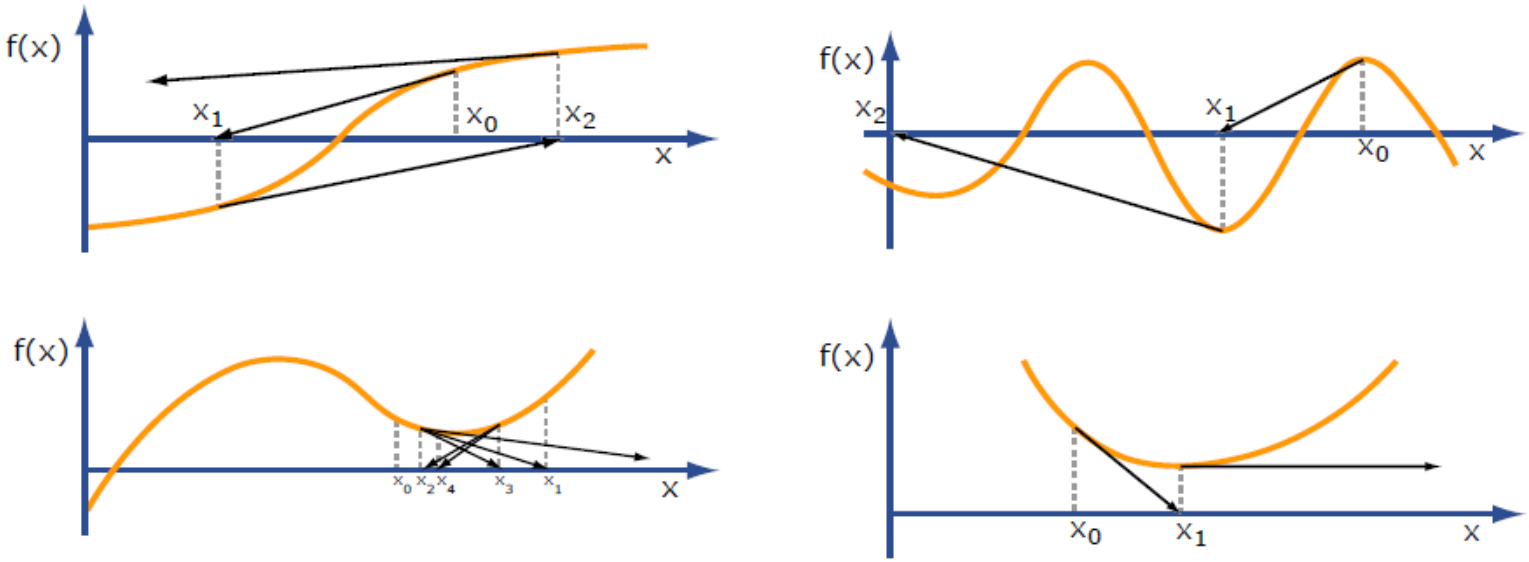
- Features:

1. Converges fast (quadratic convergence), if it is convergence.
2. Requires only one initial guess.

Open method: Newton-Raphson method

- Issues:

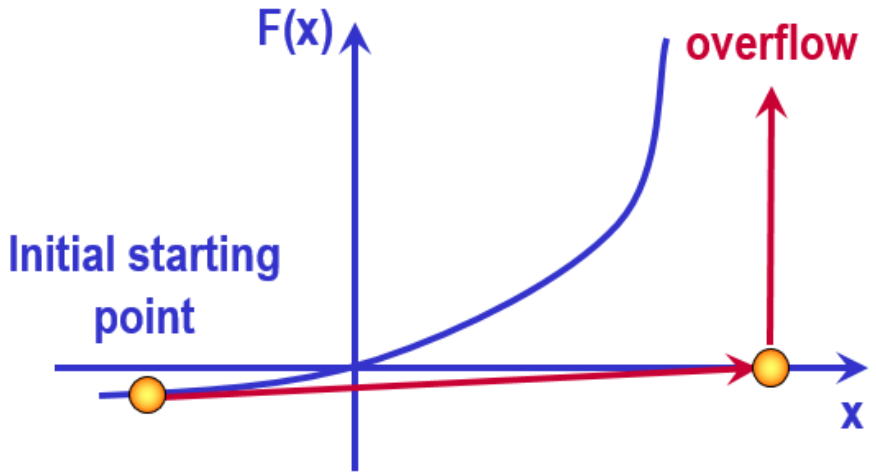
1. **Divergence**: inflection points in the vicinity of the root, i.e. *f''(x)=0*
2. **Oscillations**: Iterations can oscillate around a local minima or maxima
3. **Root Jumping**: Near-zero slope encountered
4. **Division by zero**: zero slope



Open method: Newton-Raphson method

- Modification:

1. Initial solution: Randomly select multiple initial solutions
2. Update rule: scale the step
3. Stopping criteria: set maximum iteration



$$x^{(k+1)} = x^{(k)} - \lambda \cdot \frac{F\left[x^{(k)}\right]}{F'\left[x^{(k)}\right]} \quad (0 < \lambda < 1)$$

Open method: Newton-Raphson method

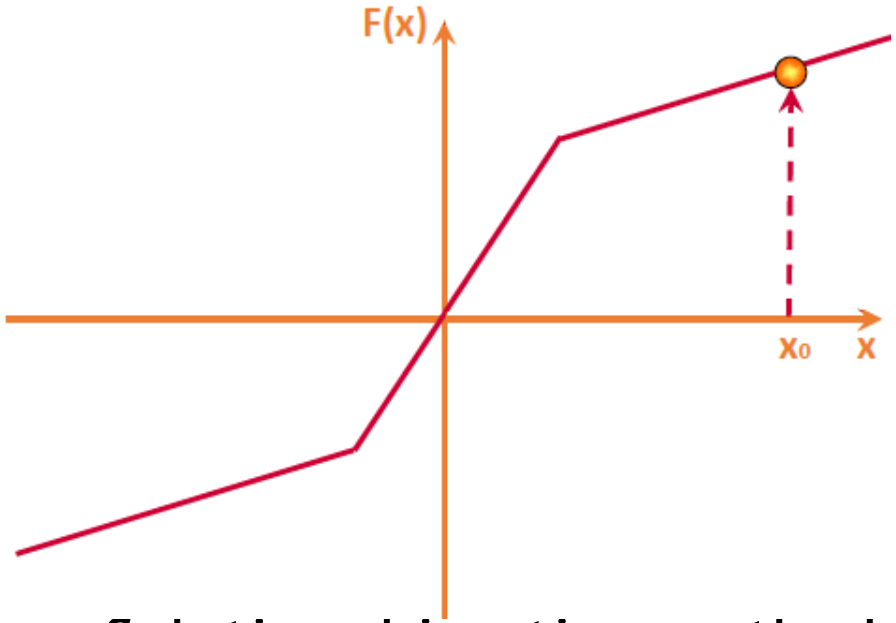
Notes in the programming:

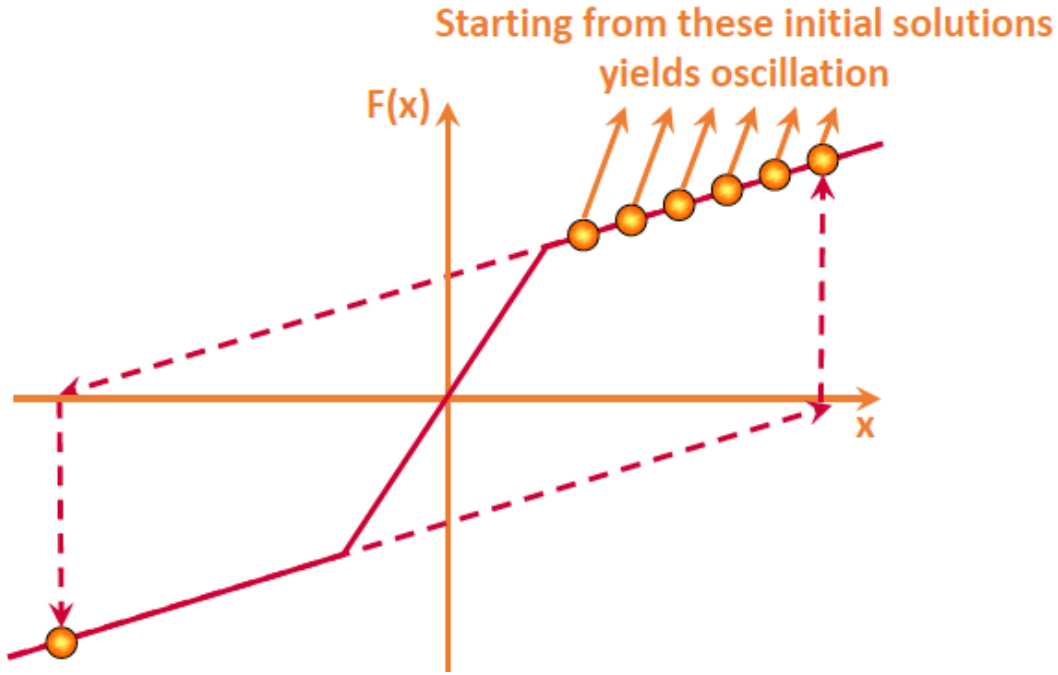
1. Always to substitute the final answer in the original function to check its result is close to zero;
2. Set upper limit on the number of iterations (i.e., set maximum iteration) guard against oscillating, slowly convergent, or divergent solutions that could persist interminably
3. Note that f’(x) might equal zero

Open method: Newton-Raphson method

Piecewise-linear function

- Question: What's the result for the initial value x0?

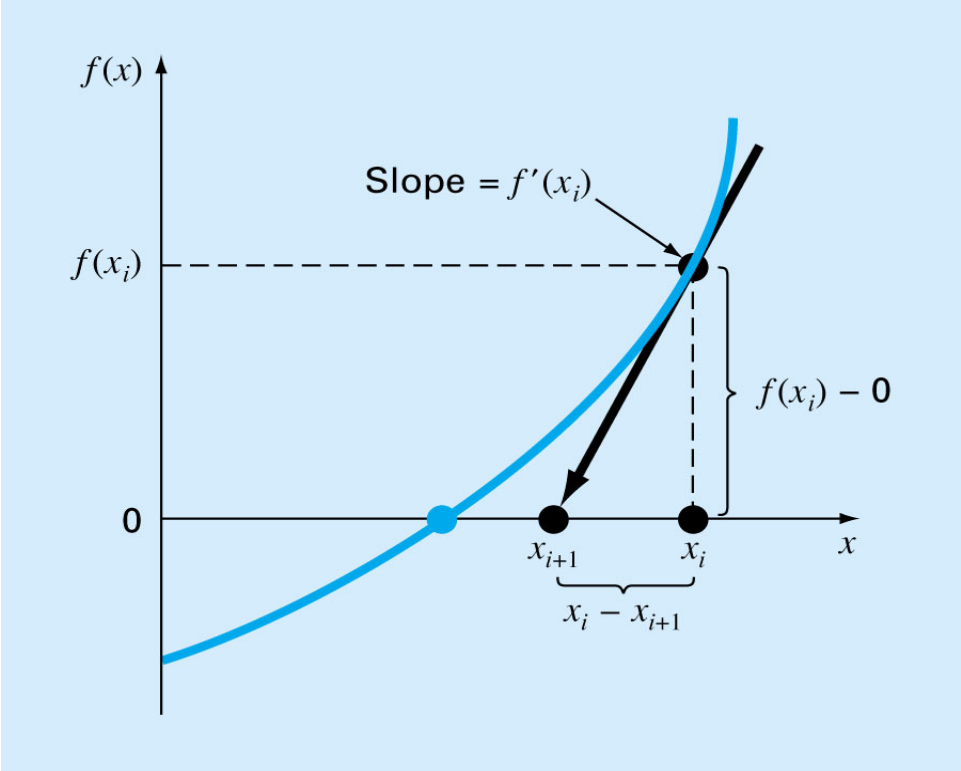




- Solution: bisection method

Open method: Newton-Raphson method

- The Newton-Raphson method is a convenient method for functions whose **derivatives can be evaluated analytically**. It may not be convenient for functions whose derivatives cannot be evaluated analytically.



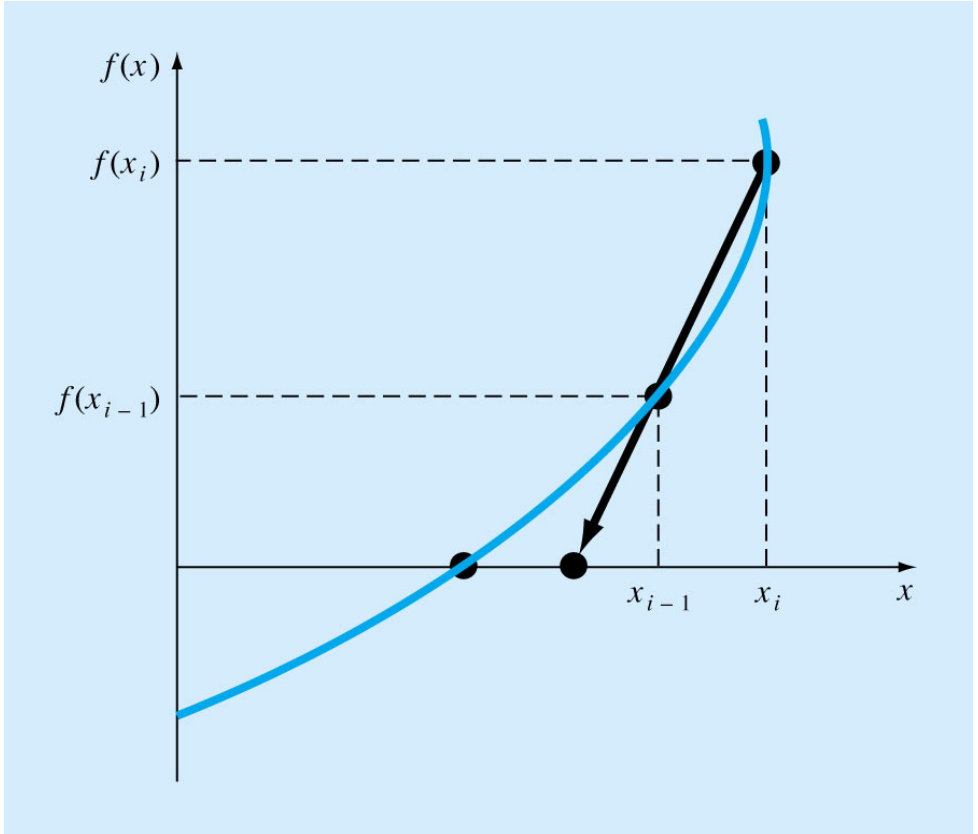
Open method: Newton-Raphson method

The Secant method

- A slight variation of Newton's method for functions whose derivatives are difficult to evaluate. For these cases the derivative can be approximated by a backward finite divided difference.

1. Requires two initial estimates of x , e.g., x_o, x_1. However, because f(x) is not required to change signs between estimates, it is not classified as a "bracketing" method.

2. The secant method has the same properties as Newton's method. **Convergence is not guaranteed for all x_o, f(x).**

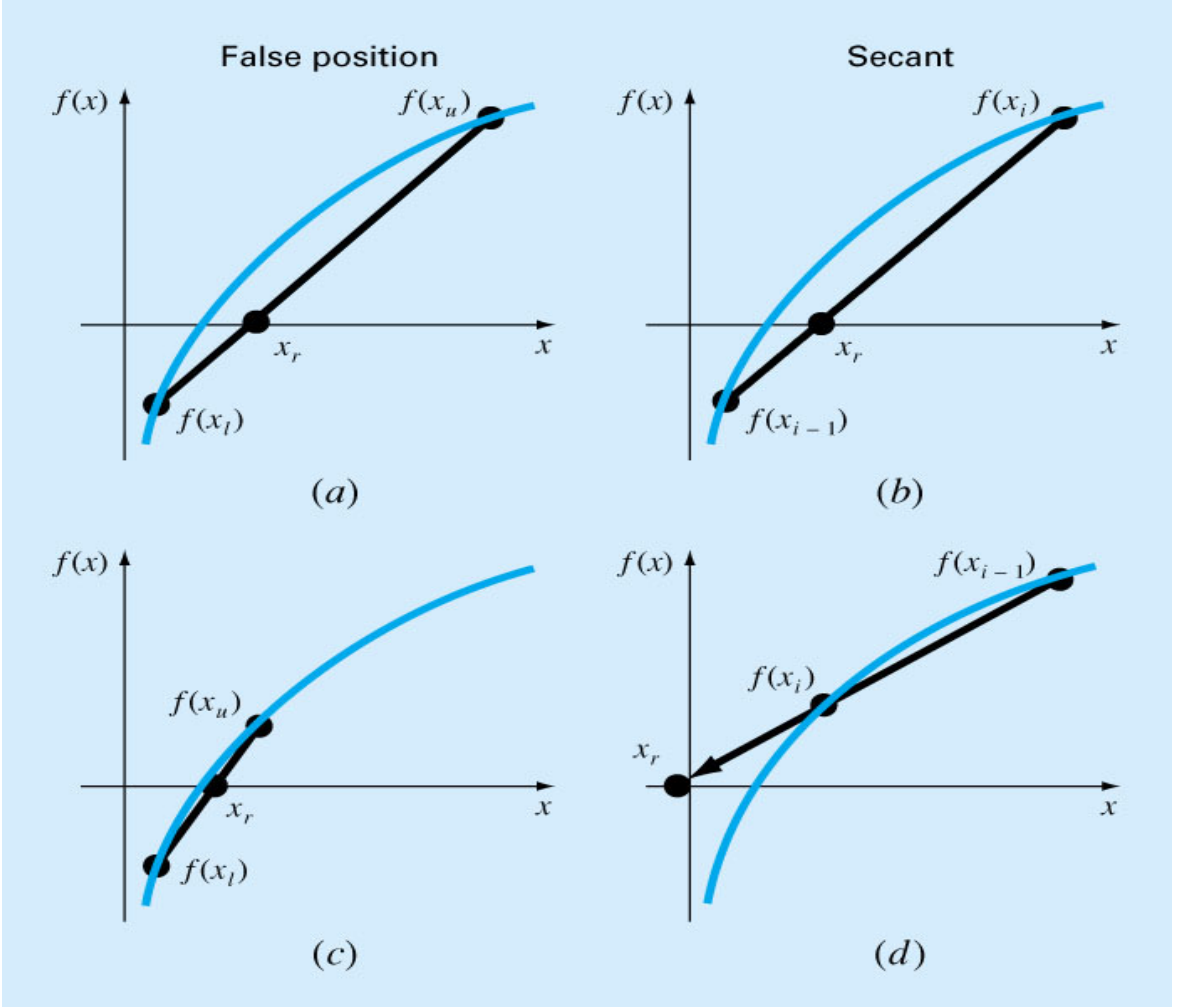


$$\frac{1}{f'(x_i)} \cong \frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})}$$

$$x_{i+1} = x_i - f(x_i)\frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})} \qquad i = 1,2,3,\ldots$$

Open method: Newton-Raphson method

- The Secant method



Open method: Bairstow's Method

- For finding a complex root of a polynomial
- For polynomials, the complex roots occur in **conjugate pairs** if the coefficients are all real-valued.
- => If we extract **the quadratic factors** that are the products of the pairs of complex roots, we can avoid complex arithmetic because such quadratic factors have real coefficients.
- => First, we need to develop the algorithm for synthetic division by **a trial quadratic**, $x^2 - rx - s$, which is hopefully near to the desired factor of the polynomial. We have,

$$P_n(x) = a_1x^n + a_2x^{n-1} + \cdots + a_{n+1} = (x^2 - rx - s)Q_{n-2}(x) + remainder$$
$$= (x^2 - rx - s)(b_1x^{n-2} + b_2x^{n-3} + \cdots + b_{n-2}x + b_{n-1}) + b_n(x - r) + b_{n+1}$$

$b_n(x - r) + b_{n+1} = 0$, if $x^2 - rx - s$ is an exact divisor of $P_n(x)$

Linear term for providing later simplicity

Open method: Bairstow's Method

$$P_n(x) = a_1x^n + a_2x^{n-1} + \cdots + a_{n+1}$$
$$= (x^2 - rx - s)(b_1x^{n-2} + b_2x^{n-3} + \cdots + b_{n-2}x + b_{n-1}) + b_n(x - r) + b_{n+1}$$

- Continued:

On multiplying out and equating coefficients, we have,

$$\begin{aligned} a_1 &= b_1 \\ a_2 &= b_2 - rb_1 \\ a_3 &= b_3 - rb_2 - sb_1 \\ a_4 &= b_4 - rb_3 - sb_2 \\ &\vdots \\ a_n &= b_n - rb_{n-1} - sb_{n-2} \\ a_{n+1} &= b_{n+1} - rb_n - sb_{n-1} \end{aligned}$$

or

$$\begin{aligned} b_1 &= a_1 \\ b_2 &= a_2 + rb_1 \\ b_3 &= a_3 + rb_2 + sb_1 \\ b_4 &= a_4 + rb_3 + sb_2 \\ &\vdots \\ b_n &= a_n + rb_{n-1} + sb_{n-2} \\ b_{n+1} &= a_{n+1} + rb_n + sb_{n-1} \end{aligned}$$

We would like b_n and b_{n+1} to be zero, for that would show $x^2 - rx - s$ to be a quadratic factor the polynomial. However, this will normally not to be true.

=>we need to properly change the values of r and s to make the remainder zero or at least make its coefficients smaller.

Open method: Bairstow's Method

- Continued:
- Expanding b_n and b_{n+1} as a Taylor series, we have,

$$b_n(r^*, s^*) = b_n(r, s) + \frac{\partial b_n}{\partial r}(r^* - r) + \frac{\partial b_n}{\partial s}(s^* - s) + \cdots,$$

$$b_{n+1}(r^*, s^*) = b_{n+1}(r, s) + \frac{\partial b_{n+1}}{\partial r}(r^* - r) + \frac{\partial b_{n+1}}{\partial s}(s^* - s) + \cdots$$

Let us take (r^*, s^*) as the point at which the remainder is zero, and,

$$r^* - r = \Delta r, s^* - s = \Delta s$$

We assume that Δr and Δs are small, so the terms in Taylor series of higher order than the first are negligible. We have,

$$b_n(r^*, s^*) = 0 \doteq b_n(r, s) + \frac{\partial b_n}{\partial r}\Delta r + \frac{\partial b_n}{\partial s}\Delta s,$$

$$b_{n+1}(r^*, s^*) = b_{n+1}(r, s) + \frac{\partial b_{n+1}}{\partial r}\Delta r + \frac{\partial b_{n+1}}{\partial s}\Delta s$$

Open method: Bairstow's Method

$$b_n(r^*, s^*) = 0 \doteq b_n(r, s) + \frac{\partial b_n}{\partial r}\Delta r + \frac{\partial b_n}{\partial s}\Delta s,$$

$$b_{n+1}(r^*, s^*) = b_{n+1}(r, s) + \frac{\partial b_{n+1}}{\partial r}\Delta r + \frac{\partial b_{n+1}}{\partial s}\Delta s$$

- Continued:
- The required partial derivatives can be obtained from the b's by a second synthetic division by the factor $x^2 - rx - s$. We can define a set of c as follows,

$$\begin{aligned} c_1 &= b_1 \\ c_2 &= b_2 + rc_1 \\ c_3 &= b_3 + rc_2 + sc_1 \\ c_4 &= b_4 + rc_3 + sc_2 \\ &\vdots \\ c_n &= b_n + rc_{n-1} + sc_{n-2} \end{aligned} \quad \Rightarrow$$

$$\frac{\partial b_1}{\partial r} = \frac{\partial a_1}{\partial r} = 0$$

$$\frac{\partial b_2}{\partial r} = r\frac{\partial b_1}{\partial r} + b_1 = b_1 = c_1$$

$$\frac{\partial b_3}{\partial r} = r\frac{\partial b_2}{\partial r} + b_2 = c_2$$

$$\frac{\partial b_4}{\partial r} = r\frac{\partial b_3}{\partial r} + b_3 + sr\frac{\partial b_2}{\partial r} = b_3 + rc_2 + sc_1 = c_3$$

$$\vdots$$

$$\frac{\partial b_n}{\partial r} = r\frac{\partial b_{n-1}}{\partial r} + b_{n-1} + sr\frac{\partial b_{n-2}}{\partial r} = b_{n-1} + rc_{n-2} + sc_{n-3} = c_{n-1}$$

Open method: Bairstow's Method

- Continued:
- Similarly, for the partial derivatives respect to s, we have,

$$\frac{\partial b_1}{\partial s} = \frac{\partial a_1}{\partial s} = 0$$

$$\frac{\partial b_2}{\partial s} = r\frac{\partial b_1}{\partial s} + \frac{\partial a_2}{\partial s} = 0$$

$$\frac{\partial b_3}{\partial s} = r\frac{\partial b_2}{\partial s} + s\frac{\partial b_1}{\partial s} + b_1 = b_1 = c_1$$

$$\frac{\partial b_4}{\partial s} = r\frac{\partial b_3}{\partial s} + s\frac{\partial b_2}{\partial s} + b_2 = b_2 + rc_1 = c_2$$

$$\vdots$$

$$\frac{\partial b_n}{\partial s} = r\frac{\partial b_{n-1}}{\partial s} + s\frac{\partial b_{n-2}}{\partial s} + b_{n-2} = b_{n-2} + rc_{n-3} + sc_{n-4} = c_{n-2}$$

Open method: Bairstow's Method

- Continued:
- The partial derivatives are equal to the properly corresponding c, and we have,

$$-b_n = c_{n-1}\Delta r + c_{n-2}\Delta s,$$
$$-b_{n+1} = c_n\Delta r + c_{n-1}\Delta s.$$

By Cramer's rule, we have,

$$\Delta r = \frac{\begin{vmatrix} -b_n & c_{n-2} \\ -b_{n+1} & c_{n-1} \end{vmatrix}}{\begin{vmatrix} c_{n-1} & c_{n-2} \\ c_n & c_{n-1} \end{vmatrix}}$$

$$\Delta s = \frac{\begin{vmatrix} c_{n-1} & -b_n \\ c_n & -b_{n+1} \end{vmatrix}}{\begin{vmatrix} c_{n-1} & c_{n-2} \\ c_n & c_{n-1} \end{vmatrix}}$$

Open method: Bairstow's Method

- Example: Find the quadratic factors of

$$x^4 - 1.1x^3 + 2.3x^2 + 0.5x + 3.3 = 0$$

Use $x^2 + x + 1$ as starting factor ($r = -1, s = -1$). We have,

	a_1	a_2	a_3	a_4	a_5
	1	-1.1	2.3	0.5	3.3
$r = -1,$		-1.0	2.1	-3.4	0.8
$s = -1$		-	-1.0	2.1	-3.4
	1	-2.1	3.4	-0.8 (b_n)	0.7 (b_{n+1})
		-1.0	3.1	-5.5	
		-	-1.0	3.1	
	1	-3.1(c_{n-2})	5.5 (c_{n-1})	-3.2 (c_n)	

$$b_1 = a_1$$
$$b_2 = a_2 + rb_1$$
$$b_3 = a_3 + rb_2 + sb_1$$
$$b_4 = a_4 + rb_3 + sb_2$$
$$\vdots$$
$$b_n = a_n + rb_{n-1} + sb_{n-2}$$
$$b_{n+1} = a_{n+1} + rb_n + sb_{n-1}$$

$$c_1 = b_1$$
$$c_2 = b_2 + rc_1$$
$$c_3 = b_3 + rc_2 + sc_1$$
$$c_4 = b_4 + rc_3 + sc_2$$
$$\vdots$$
$$c_n = b_n + rc_{n-1} + sc_{n-2}$$

Open method: Bairstow's Method

- Example: continued

$$\Delta r = \frac{\begin{vmatrix} -b_n & c_{n-2} \\ -b_{n+1} & c_{n-1} \end{vmatrix}}{\begin{vmatrix} c_{n-1} & c_{n-2} \\ c_n & c_{n-1} \end{vmatrix}} = \frac{\begin{vmatrix} 0.8 & -3.1 \\ -0.7 & 5.5 \end{vmatrix}}{\begin{vmatrix} 5.5 & -3.1 \\ -3.2 & 5.5 \end{vmatrix}} = \frac{2.23}{20.33} = 0.11, \Rightarrow r^* = -1 + 0.11 = -0.89$$

$$\Delta s = \frac{\begin{vmatrix} c_{n-1} & -b_n \\ c_n & -b_{n+1} \end{vmatrix}}{\begin{vmatrix} c_{n-1} & c_{n-2} \\ c_n & c_{n-1} \end{vmatrix}} = \frac{\begin{vmatrix} 5.5 & 0.8 \\ -3.2 & -0.7 \end{vmatrix}}{\begin{vmatrix} 5.5 & -3.1 \\ -3.2 & 5.5 \end{vmatrix}} = -\frac{1.29}{20.33} = -0.06, \Rightarrow s^* = -1.06$$

Open method: Bairstow's Method

- Example: Continued

Second round:

	a_1	a_2	a_3	a_4	a_5
	1	-1.1	2.3	0.5	3.3
$r = -0.89,$		-0.89	1.77	-2.68	0.06
$s = -1.06$		-	-1.06	2.11	-3.17
	1	-1.99	3.01	-0.07 (b_n)	0.17 (b_{n+1})
		-0.89	2.56	-4.01	
		-	-1.06	3.05	
	1	-1(c_{n-2})	-2.88 (c_{n-1})	4.51 (c_n)	-1.03

$$b_1 = a_1$$
$$b_2 = a_2 + rb_1$$
$$b_3 = a_3 + rb_2 + sb_1$$
$$b_4 = a_4 + rb_3 + sb_2$$
$$\vdots$$
$$b_n = a_n + rb_{n-1} + sb_{n-2}$$
$$b_{n+1} = a_{n+1} + rb_n + sb_{n-1}$$

$$c_1 = b_1$$
$$c_2 = b_2 + rc_1$$
$$c_3 = b_3 + rc_2 + sc_1$$
$$c_4 = b_4 + rc_3 + sc_2$$
$$\vdots$$
$$c_n = b_n + rc_{n-1} + sc_{n-2}$$

Open method: Bairstow's Method

- Example: continued

$$\Delta r = \frac{\begin{vmatrix} -b_n & c_{n-2} \\ -b_{n+1} & c_{n-1} \end{vmatrix}}{\begin{vmatrix} c_{n-1} & c_{n-2} \\ c_n & c_{n-1} \end{vmatrix}} = \frac{\begin{vmatrix} 0.07 & -2.88 \\ -0.17 & 4.51 \end{vmatrix}}{\begin{vmatrix} 4.51 & -2.88 \\ -1.03 & 4.51 \end{vmatrix}} = \frac{-0.175}{17.374} = -0.010, \Rightarrow r^* = -0.89 - 0.010 = -0.9$$

$$\Delta s = \frac{\begin{vmatrix} c_{n-1} & -b_n \\ c_n & -b_{n+1} \end{vmatrix}}{\begin{vmatrix} c_{n-1} & c_{n-2} \\ c_n & c_{n-1} \end{vmatrix}} = \frac{\begin{vmatrix} 4.51 & 0.07 \\ -1.03 & -0.17 \end{vmatrix}}{\begin{vmatrix} 4.51 & -2.88 \\ -1.03 & 4.51 \end{vmatrix}} = \frac{-0.694}{17.374} = -0.040, \Rightarrow s^* = -1.10$$

The exact factors are $(x^2 0.9x + 1.1)(x^2 - 2x + 3)$.

Open method: Quotient-Difference Algorithm

- All the methods discussed so far require a starting point sufficient near to the root or to the quadratic factor being sought.
- Q-D algorithm is a relatively efficient method to determine all roots of polynomial without starting points.
- For the given polynomials,

$$P_n(x) = a_1 x^n + a_2 x^{n-1} + \cdots + a_{n+1},$$

We can form an array of q and e terms, starting the tableau by calculating a first row q and a second row of e,

$$q^{(1)} = -\frac{a_2}{a_1}, all\ other\ q's\ are\ zero,$$

$$e^{(i)} = \frac{a_{i+2}}{a_{i+1}}, i = 1, 2, 3, \cdots, n-1,$$

$$e^{(0)} = e^{(n)} = 0$$

Open method: Quotient-Difference Algorithm

- Continued:
- The start of the array is

$e^{(0)}$	$q^{(1)}$	$e^{(1)}$	$q^{(2)}$	$e^{(2)}$	$q^{(3)}$	$\cdots$	$e^{(n-1)}$	$q^{(n)}$	$e^{(n)}$
	$-\frac{a_2}{a_1}$		0		0			0	
0		$\frac{a_3}{a_2}$		$\frac{a_4}{a_3}$			$\frac{a_{n+1}}{a_n}$		0

- A new row of q's is computed by,

$$new\ q^{(i)} = e^{(i)} - e^{(i-1)} + q^{(i)}$$

Note: the algorithm is "e" to right minus "e" to left plus q above

Open method: Quotient-Difference Algorithm

- Continued:
- The start of the array is

$e^{(0)}$	$q^{(1)}$	$e^{(1)}$	$q^{(2)}$	$e^{(2)}$	$q^{(3)}$	$\cdots$	$e^{(n-1)}$	$q^{(n)}$	$e^{(n)}$
	$-\frac{a_2}{a_1}$		0		0			0	
0		$\frac{a_3}{a_2}$		$\frac{a_4}{a_3}$			$\frac{a_{n+1}}{a_n}$		0

- A new row of e's is computed by,

$$new\ e^{(i)} = \left(\frac{q^{(i+1)}}{q^{(i)}}\right)e^{(i)}$$

Note: the algorithm is that "q" to right over "q" to left times e above.

Open method: Quotient-Difference Algorithm

- Example: $P_4(x) = 128x^4 - 256x^3 + 160x^2 - 32x + 1$
- The table is

$e^{(0)}$	$q^{(1)}$	$e^{(1)}$	$q^{(2)}$	$e^{(2)}$	$q^{(3)}$	$e^{(3)}$	$q^{(4)}$	$e^{(4)}$
	$-\frac{a_2}{a_1} = 2.00$		0		0		0	
0		$\frac{a_3}{a_2} = -0.625$		$\frac{a_4}{a_3} = -0.200$		$\frac{a_{n+1}}{a_n} = \frac{1}{-32} = 0.031$		0
	$e^{(i)} - e^{(i-1)} + q^{(i)} = -0.625 - 0 + 2 = 1.375$		0.425		0.169			
0		$\left(\frac{q^{(i+1)}}{q^{(i)}}\right) e^{(i)} = \left(\frac{0.425}{1.375}\right) \times -0.625 = -0.193$						0

Open method: Quotient-Difference Algorithm

- Example: $P_4(x) = 128x^4 - 256x^3 + 160x^2 - 32x + 1$
- Continued: Continuing to compute rows of q's and then e's until all the e-values approach zero. Then, the q-values assume the value of the roots.
- (the 9th round)

$e^{(0)}$	$q^{(1)}$	$e^{(1)}$	$q^{(2)}$	$e^{(2)}$	$q^{(3)}$	$e^{(3)}$	$q^{(4)}$	$e^{(4)}$
	0.980		0.673		0.307		0.038	
0		−0.005		−0.001		−0.000		0
	0.975		0.677		0.308		0.308*	

- The true values of the roots are 0.96194, 0.69134, 0.30866, and 0.03806

Open method: Quotient-Difference Algorithm

- The Q-D algorithm can also solve the problem with conjugate complex roots.
- Technically, if the given problem contains conjugate complex roots, we will observe that **the e's will not approach zero but will fluctuate in value**.
- In this case, the sum of two q-values on either side of this e will approach r and the product of the q above and to the left times the q below and to the right approaches "-s" in the factor $(x^2 - rx - s)$.

Open method: Quotient-Difference Algorithm

- Example: $P(x) = x^4 - 6x^3 + 12x^2 - 19x + 12$

$e^{(0)}$	$q^{(1)}$	$e^{(1)}$	$q^{(2)}$	$e^{(2)}$	$q^{(3)}$	$e^{(3)}$	$q^{(4)}$	$e^{(4)}$
	6.000		0		0		0	
0		-2.000		-1.583		-0.632		0
	4.000		0.417		0.951		0.632	
0		-0.208		-3.610		-0.420		0
	3.792		-2.985		4.141		1.052	
0		0.164		5.008		-0.107		0

$$q^{(1)} = -\frac{a_2}{a_1}, all\ other\ q's\ are\ zero,$$

$$e^{(i)} = \frac{a_{i+2}}{a_{i+1}}, i = 1, 2, 3, \cdots, n-1,$$

$$e^{(0)} = e^{(n)} = 0$$

$$new\ q^{(i)} = e^{(i)} - e^{(i-1)} + q^{(i)}$$

$$new\ e^{(i)} = (\frac{q^{(i+1)}}{q^{(i)}})e^{(i)}$$

Open method: Quotient-Difference Algorithm

- Example: Continued (6th round)
- $P(x) = x^4 - 6x^3 + 12x^2 - 19x + 12$

$e^{(0)}$	$q^{(1)}$	$e^{(1)}$	$q^{(2)}$	$e^{(2)}$	$q^{(3)}$	$e^{(3)}$	$q^{(4)}$	$e^{(4)}$
	4.017		4.712		-3.687		0.958	
0		-0.019		-4.333		-0.019		0
	3.998		0.398		0.627		0.977	
0		-0.002		-6.826		-0.030		0
	4.000		-6.426		7.423		1.007	
0		0.003		7.885		-0.004		0
	4.003		1.456		-0.466		1.010	

Fluctuation => contains conjugate complex roots.

$$q^{(1)} = -\frac{a_2}{a_1}, all\ other\ q's\ are\ zero,$$
$$e^{(i)} = \frac{a_{i+2}}{a_{i+1}}, i = 1, 2, 3, \cdots, n-1,$$
$$e^{(0)} = e^{(n)} = 0$$
$$new\ q^{(i)} = e^{(i)} - e^{(i-1)} + q^{(i)}$$
$$new\ e^{(i)} = (\frac{q^{(i+1)}}{q^{(i)}})e^{(i)}$$

Open method: Quotient-Difference Algorithm

- Example: Continued (8th round)
- $P(x) = x^4 - 6x^3 + 12x^2 - 19x + 12$

the sum of two q-values on either side of this e will approach r and the product of the q above and to the left times the q below and to the right approaches "-s"

$e^{(0)}$	$q^{(1)}$	$e^{(1)}$	$q^{(2)}$	$e^{(2)}$	$q^{(3)}$	$e^{(3)}$	$q^{(4)}$	$e^{(4)}$
0		-0.002		-6.826		-0.030		0
	4.000		-6.426		7.423		1.007	
0		0.003		7.885		-0.004		0
	4.003		1.456		-0.466		1.010	

The result of the Q-D algorithm for the polynomial is $(x-1)(x-4)(x^2-x+3) = x^4 - 6x^3 + 12x^2 - 19x + 12$, i.e., $q^{(1)}$ converging to 4, $q^{(4)}$ converging to 1. Since $e^{(2)}$ does not approach zero, $q^{(2)}$ and $q^{(3)}$ represent a quadratic factor:

$$r \doteq q^{(2)} + q^{(3)} = 1.456 - 0.466 = 0.990;$$
$$s \doteq -(-6.426)\times(-0.466) = -2.995$$

Open method: Quotient-Difference Algorithm

- Example: Continued

$$P(x) = x^4 - 6x^3 + 12x^2 - 19x + 12$$

$$r \doteq q^{(2)} + q^{(3)} = 1.456 - 0.466 = 0.990;$$

$$s \doteq -(-6.426)\times(-0.466) = -2.995$$

- The quadratic factor is $x^2 - rx - s = x^2 - 0.990x + 2.995 \approx x^2 - x + 3$
- Two equal roots behave similarly.

Open method: Quotient-Difference Algorithm

- Pitfall of Q-D algorithm?
- **Division by zero.**
- We cannot compute the first q and e rows if one of the coefficients in the polynomial is zero.

- In such a case, we change the variable to $y = x - 1$.
- Note: Subtracting 1 from the roots of the equation is an arbitrary choice, but this facilitates the reverse change of variable to get the roots of the original equation after the roots of the new equation in y has been found.

Open method: Quotient-Difference Algorithm

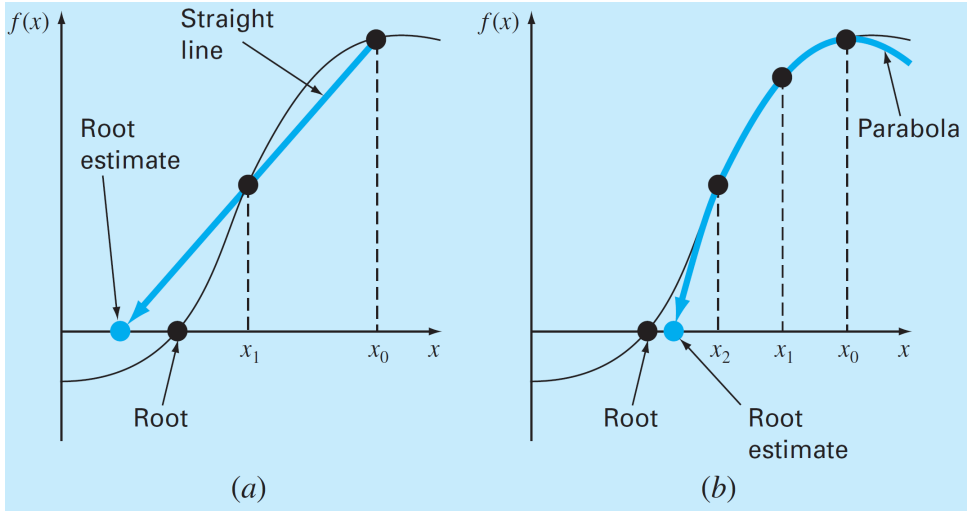
- Example: $f(x) = x^4 - 2x^2 + x - 1 = 0$
- We let y=x-1 and use repeated synthetic division to determine the coefficients of f(y)=0. The successive remainder on dividing by x-1 are the coefficients of f(y).

					1
1	0	-2	1	-1	
	1	1	-1	0	
1	1	-1	0	-1	
	1	2	1		
1	2	1	1		
	1	3			
1	3	4			
	1				
1	4				
1					

Therefore, we have, $f(y) = y^4 + 4y^3 + 4y^2 + y - 1$. Now, we can use Q-D algorithm to solve the problem and then get the roots of f(x)=0 by adding 1.

Open method: Muller's Method

- Muller's method is an interpolation method that uses **quadratic interpolation** rather than linear.
- => A second-degree polynomial is made to fit **three points near a root**, i.e., $[x_0, f(x_0)], [x_1, f(x_1)], [x_2, f(x_2)]$, and **the proper zero** of this quadratic, using the quadratic formula, is used as the improved estimate of the root.
- => The process is then repeated using the set of three points nearest the root being evaluated.



Open method: Muller's Method

- The procedure for Muller's method is developed by writing a quadratic equation that fits through three points in the vicinity of a root $[x_0, f(x_0)], [x_1, f(x_1)], [x_2, f(x_2)]$, in the form,

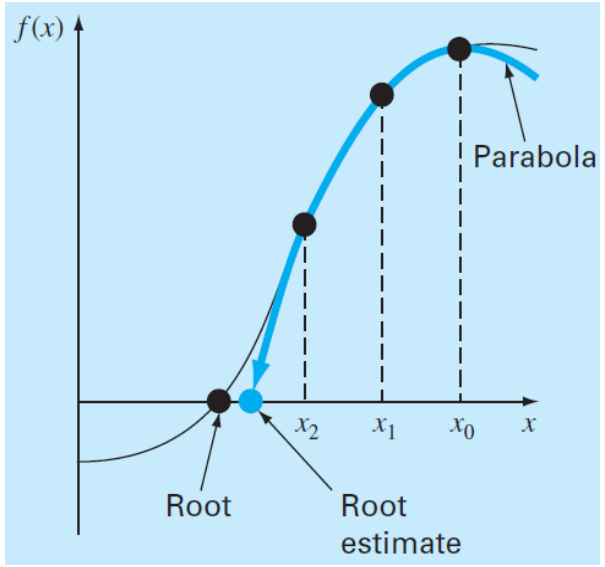
$$av^2 + bv + c$$

For simplicity, we can write parabolic equation in a convenient form (axes passing through $[x_2, f(x_2)]$), i.e.,

$$f(x) = a(x - x_2)^2 + b(x - x_2) + c$$

The coefficients can be evaluated by substituting each of the three points

$$f(x_0) = a(x_0 - x_2)^2 + b(x_0 - x_2) + c$$
$$f(x_1) = a(x_1 - x_2)^2 + b(x_1 - x_2) + c$$
$$f(x_2) = a(x_2 - x_2)^2 + b(x_2 - x_2) + c$$



Open method: Muller's Method

- Continued:

$$\left.\begin{aligned} f(x_0) &= a(x_0 - x_2)^2 + b(x_0 - x_2) + c \\ f(x_1) &= a(x_1 - x_2)^2 + b(x_1 - x_2) + c \\ f(x_2) &= a(x_2 - x_2)^2 + b(x_2 - x_2) + c \end{aligned}\right\} \Rightarrow \begin{cases} c = f(x_2) \\ f(x_0) - f(x_2) = a(x_0 - x_2)^2 + b(x_0 - x_2) \ (1) \\ f(x_1) - f(x_2) = a(x_1 - x_2)^2 + b(x_1 - x_2) \end{cases}$$

Here, we define,

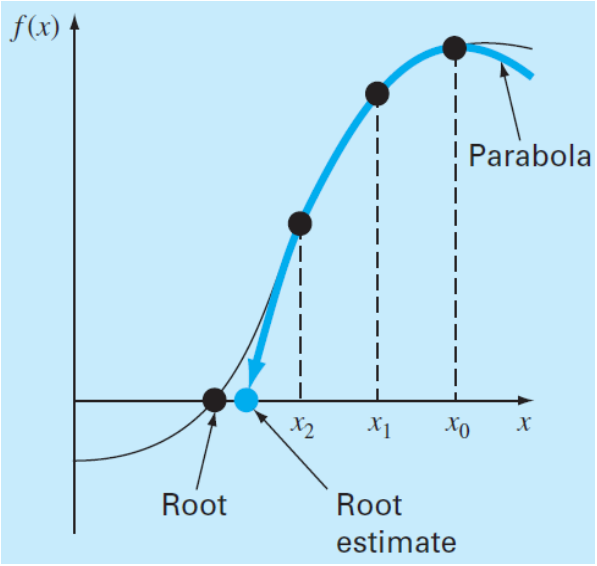
$$h_0 = x_1 - x_0, h_1 = x_2 - x_1, \delta_0 = \frac{f(x_1) - f(x_0)}{x_1 - x_0}, \delta_1 = \frac{f(x_2) - f(x_1)}{x_2 - x_1}$$

Substituting these to Eq. 1, we have,

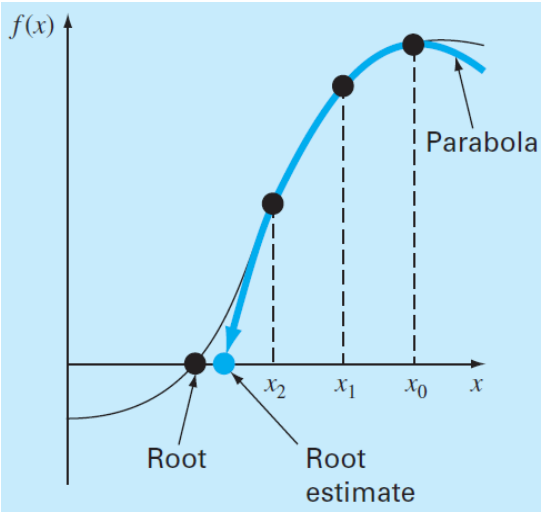
$$(h_0 + h_1)b - (h_0 + h_1)^2 a = h_0\delta_0 + h_1\delta_1,$$
$$h_1 b - h_1^2 a = h_1\delta_1$$

Then, we have,

$$a = \frac{\delta_1 - \delta_0}{h_1 + h_0}$$
$$b = ah_1 + \delta_1$$
$$c = f(x_2)$$



Open method: Muller's Method



- Continued: $f(x) = a(x - x_2)^2 + b(x - x_2) + c$
- Now we have,

$$a = \frac{\delta_1 - \delta_0}{h_1 + h_0}$$
$$b = ah_1 + \delta_1$$
$$c = f(x_2)$$

Then, by using the quadratic formula, we can derive the estimated root as follows,

$$x_3 - x_2 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-2c}{b \pm \sqrt{b^2 - 4ac}}$$

Note that the use of the quadratic formula means that both real and complex roots can be located. Here, we select the root closest to x_2.

Open method: Muller's Method

- Example: Use Muller's method with guesses of $x_0, x_1, and\ x_2 = 4.5, 5.5, and\ 5$, respectively, to determine a root of the equation

$$f(x) = x^3 - 13x - 12$$

Solution: First, we evaluate the function at the guesses,

$$f(x_0 = 4.5) = 20.625, f(x_1 = 5.5) = 82.875, f(x_2 = 5) = 48$$

Then, we derive,

$$h_0 = x_1 - x_0 = 1, \qquad h_1 = x_2 - x_1 = -0.5,$$

$$\delta_0 = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = 62.25, \qquad \delta_1 = \frac{f(x_2) - f(x_1)}{x_2 - x_1} = 69.75$$

Accordingly, we have,

$$a = \frac{\delta_1 - \delta_0}{h_1 + h_0} = 15$$

$$b = ah_1 + \delta_1 = 62.25$$

$$c = f(x_2) = 48$$

Open method: Muller's Method

- Example: Use Muller's method with guesses of $x_0, x_1, and\ x_2 = 4.5, 5.5, and\ 5$, respectively, to determine a root of the equation

$$f(x) = x^3 - 13x - 12$$

Solution: First, we evaluate the function at the guesses,

$$f(x_0 = 4.5) = 20.625, f(x_1 = 5.5) = 82.875, f(x_2 = 5) = 48$$

Then, we derive,

$$h_0 = x_1 - x_0 = 1, \qquad h_1 = x_2 - x_1 = -0.5,$$

$$\delta_0 = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = 62.25, \qquad \delta_1 = \frac{f(x_2) - f(x_1)}{x_2 - x_1} = 69.75$$

Accordingly, we have,

$$a = \frac{\delta_1 - \delta_0}{h_1 + h_0} = 15$$

$$b = ah_1 + \delta_1 = 62.25$$

$$c = f(x_2) = 48$$

$$\Rightarrow x_3 = x_2 + \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = 3.976487$$

Open method: Muller's Method

- Example: Use Muller's method with guesses of $x_0, x_1, and\ x_2 = 4.5, 5.5, and\ 5$, respectively, to determine a root of the equation

$$f(x) = x^3 - 13x - 12$$

Solution (continued): Check stopping criterion,

$$\epsilon_a = \left|\frac{x_3 - x_2}{x_3}\right| \times 100\% = 25.74\%$$

The error is large, new guesses are assigned, i.e., $x_0 = x_1 = 5.5, x_1 = x_2 = 5, and\ x_2 = x_3 = 3.976487$; the calculation is repeated, and we have,

i	x_r	$\epsilon_a(\%)$
0	5	
1	3.976487	25.74
2	4.00105	0.6139
3	4	0.0262
4	4	0.0000119

Summary

- Both Bisection and False Position (including modified version)
  - Convergent methods, but can be slow (linear) and thus can require lots of function evaluations, each of which can be very expensive
- For Newton-Raphson, Converges fast (quadratic convergence), if it converges; but no general convergence criterion.
- The behaviors for root finding methods cannot be generalized (results are function dependent)
- Methods for finding roots of polynomials: Bairstow's method, Q-D algorithm, Muller's method.
- Always substitute estimate of root x_r at the end to check how close estimate is to $f(x_r) = 0$.