MTS2A3
Komputer & Simulasi

Pemodelan Matematik (Lanjutan)

Dr. Nurly Gofar

Program Studi Teknik Sipil

Program Pascasarjana

Universitas Bina Darma Palembang

TAYLOR'S THEOREM

If a function *f* and its first *n+1* derivatives are continuous on an interval containing *a* and *x*, then the value of the function at *x* is given by:

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(x)}{2!}(x-a)^2$$

$$+\frac{f''(x)}{3!}(x-a)^3 + \mathrm{K}$$

$$+\frac{f^n(x)}{n!}(x-a)^n + R_n$$

Where the remainder R_n is defined as:

$$R_n = \int_a^x \frac{f^{(n)}(x-t)^n}{n!} f^{(n+1)}(t)dt$$

Taylor's Series (1)

The Taylor series provides a means to predict the function value at one point in terms of the function values and its derivatives at another point.

The theorem states that any smooth function can be approximated as a polynomial.

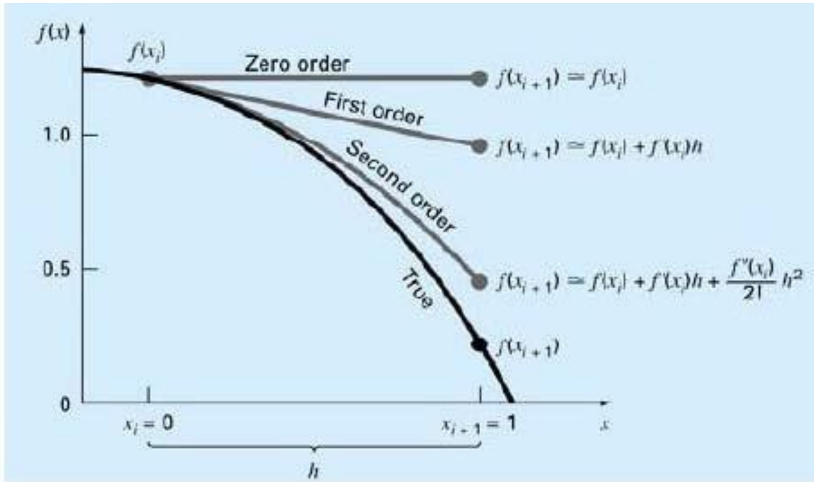
$$f(x_{i+1}) = f(x_i) + f'(x_i)(x_{i+1} - x_i) + \frac{f''(x_i)}{2!}(x_{i+1} - x_i)^2 + \cdots + \frac{f^{(n)}(x_i)}{n!}(x_{i+1} - x_i)^n + \cdots$$

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \cdots + \frac{f^{(n)}(x_i)}{n!}h^n + \cdots$$

$$h = x_{i+1} - x_i$$

$$f(x_{i+1}) = \sum_{j=0}^{\infty} \frac{f^{(j)}(x_i)}{j!}h^j$$

Function involving transcendental and trigonometric function will require an infinite number of terms in the Taylor Series to obtain accurate solution.



Taylor's Series (2)

The infinite Taylor series can also be written as

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \cdots + \frac{f^{(n)}(x_i)}{n!}h^n + R_n$$

The remainder term R_n is to account for all terms from $n+1$ to infinity:

$$R_n = \frac{f^{(n+1)}(\xi)}{(n+1)!}h^{n+1} \leftarrow R_n = \frac{f^{(n+1)}(x_i)}{(n+1)!}h^{n+1} + \frac{f^{(n+2)}(x_i)}{(n+2)!}h^{n+2} + \cdots$$

ξ = a value of *x* that lies somewhere between x_i and x_{i+1}

In most cases, the inclusion of only a few terms will result in an approximation that is close enough to the true value. The assessment of how many terms are required to get "close enough" is based on R_n.

$$R_n = O(h^{n+1})$$

$O(h^{n+1})$ means that the truncation error is of the order of h^{n+1}. For example, if the error is $O(h^2)$, *halving the step size h* will *quarter the error*.

Example 2

Use Taylor series expansions with n = 0 to 3 to approximate $f(x)$ = cos x at x_{i+1} = π/3 on the basis of the value of $f(x)$ and its derivatives at x_i = π/4.

Solution:

The exact value: $f(\pi / 3) = \cos(\pi / 3) = 0.5$

$$h = \frac{\pi}{3} - \frac{\pi}{4} = \frac{\pi}{12}$$

The zero-order approximation:

$$f(\frac{\pi}{3}) \cong f(\frac{\pi}{4}) = \cos(\frac{\pi}{4}) = 0.707106781$$

$$\varepsilon_t = \left|\frac{0.5 - 0.707106781}{0.5}\right| \times 100\% = 41.4\%$$

The first-order approximation:

$$f(\frac{\pi}{3}) \cong \cos(\frac{\pi}{4}) - \sin(\frac{\pi}{4})(\frac{\pi}{12}) = 0.521986659 \qquad \varepsilon_t = 4.40\%$$

Example 2 (cont'd)

The second-order approximation:

$$f(\frac{\pi}{3}) \cong 0.521986659 - \frac{\cos(\pi/4)}{2}(\frac{\pi}{12})^2 = 0.497754491$$

$$\varepsilon_t = 0.449\%$$

The third-order approximation:

$$f(\frac{\pi}{3}) \cong 0.497754491 + \frac{\sin(\pi/4)}{6}(\frac{\pi}{12})^3 = 0.499869147$$

$$\varepsilon_t = 0.0262\%$$

Although the addition of more terms will reduce the error further, the improvement becomes negligible.

Using Taylor Series to Estimate Truncation Errors

The remainder term R_n is to account for all terms from n+1 to infinity:

$$R_n = \int_a^x \frac{f^{(n)}(x-t)^n}{n!} f^{(n+1)}(t)dt$$

By Mean Value Theorem, it can also be written as:

$$R_n = \frac{f^{(n+1)}(\xi)}{(n+1)!} h^{n+1}$$

Note that:

ξ lies between x_i and x_{i+1}

$f^{(n+1)}$ is usually not known

But we can control h, and hence have some control over the error (remainder) associated with the truncated Taylor's polynomial.

Taylor's Series (3)

Recall the previous example of falling parachutist. The velocity at time t_{i+1} can be expressed in *Taylor series*:

$$v(t_{i+1}) = v(t_i) + v'(t_i)(t_{i+1} - t_i) + \frac{v''(t_i)}{2!}(t_{i+1} - t_i)^2 + \cdots$$

$$v(t_{i+1}) = v(t_i) + v'(t_i)(t_{i+1} - t_i) + R_1$$

$$v'(t_i) = \underbrace{\frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i}}_{\text{First-order approximation}} - \underbrace{\frac{R_1}{t_{i+1} - t_i}}_{\text{Truncation error}}$$

The truncation error associated with first-order approximation:

$$\frac{R_1}{t_{i+1} - t_i} = \frac{v''(\xi)}{2!}(t_{i+1} - t_i) = O(t_{i+1} - t_i)$$

The error of the derivative approximation is proportional to the step size. *Halving the step size* will *halve the error* of the derivative.

Example 3

Employ the first-order Taylor series to approximate $f(x) = x^4$ for various values of step size h. Take $x_i = 1$.

Solution:

$$f(x_{i+1}) \cong f(x_i) + f'(x_i)h = x_i^4 + 4x_i^3 h = 1 + 4h$$

$$R_1 = \frac{f''(x_i)}{2!}h^2 + \frac{f^{(3)}(x_i)}{3!}h^3 + \frac{f^{(4)}(x_i)}{4!}h^4 = 6h^2 x_i^2 + 4h^3 x_i + h^4 = 6h^2 + 4h^3 + h^4$$

h	X_{i+1}	TRUE $f(X_{i+1})$	Ist order	R_1	Red. Of h	Red. Of R_1
1.0000000000	2.0000000000	16.0000000000	5.0000000000	11.0000000000		
0.5000000000	1.5000000000	5.0625000000	3.0000000000	2.0625000000	0.50	0.19
0.2500000000	1.2500000000	2.4414062500	2.0000000000	0.4414062500	0.50	0.21
0.1250000000	1.1250000000	1.6018066406	1.5000000000	0.1018066406	0.50	0.23
0.0625000000	1.0625000000	1.2744293213	1.2500000000	0.0244293213	0.50	0.24
0.0312500000	1.0312500000	1.1309823990	1.1250000000	0.0059823990	0.50	0.24
0.0156250000	1.0156250000	1.0639801621	1.0625000000	0.0014801621	0.50	0.25
0.0078125000	1.0078125000	1.0316181220	1.0312500000	0.0003681220	0.50	0.25
0.0039062500	1.0039062500	1.0157167914	1.0156250000	0.0000917914	0.50	0.25
0.0019531250	1.0019531250	1.0078354180	1.0078125000	0.0000229180	0.50	0.25
0.0009765625	1.0009765625	1.0039119758	1.0039062500	0.0000057258	0.50	0.25
0.0004882813	1.0004882813	1.0019545560	1.0019531250	0.0000014310	0.50	0.25
0.0002441406	1.0002441406	1.0009769202	1.0009765625	0.0000003577	0.50	0.25

As $R_1 = O(h^2)$, the *error will be quartered if h is halved* for sufficiently small h.

Numerical Differentiation - First Derivatives

The first order expression of the derivative obtained from Taylors series expansion may be written as:

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

$$or \quad f'(x_i) = \frac{\otimes f_i}{h} + O(h)$$

Finite divide difference

Taylor Series can also be expanded backward to calculate a previous value, leading to the formulation of first backward difference

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

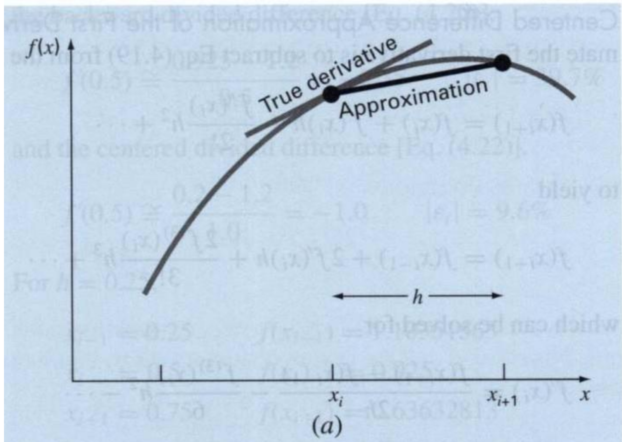
Truncation error *O(h)*

The centred difference is given as:

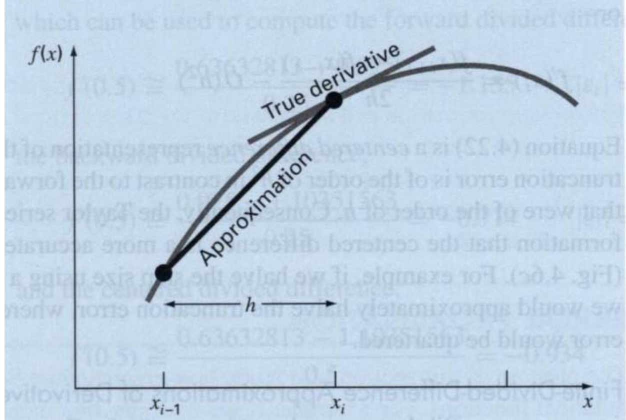
$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i - 1)}{2h_i} - O(h^2)$$

Truncation error is of $O(h^2)$

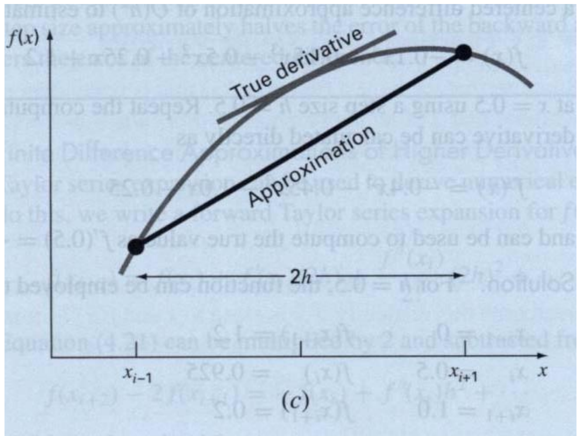
Graphical Depiction Forward, Backward and Centered Finite Difference



Forward Difference



Backward Difference



Central Difference

Stability and Conditioning

The condition of a mathematical formulation relates to its sensitivity to changes in its input values.

We say a computation is numerically unstable if the uncertainty in the input values is grossly magnified by the numerical method.

Using first-order Taylor series $f(x) = f(\tilde{x}) + f'(\tilde{x})(x - \tilde{x})$

Then, relative error: $\frac{f(x) - f(\tilde{x})}{f(x)} \cong \frac{f'(\tilde{x})(x - \tilde{x})}{f(\tilde{x})}$

The relative error of x is: $\frac{(x - \tilde{x})}{\tilde{x}}$

and a condition number can be defined as: $\frac{\tilde{x} f'(\tilde{x})}{f(\tilde{x})}$

Condition number < 1 indicate that the numerical method is well-attenuated. A condition number much greater than 1 shows the method is very ill-conditioned.

Example

Compute and interpret the condition number for f(x) = tan(x) for $\tilde{x} = \frac{\pi}{2} \pm 0.1(\frac{\pi}{2})$

Solution:

Condition number $= \dfrac{\tilde{x}(1/\cos^2 x)}{\tan \tilde{x}}$

For $\tilde{x} = \frac{\pi}{2} + 0.1(\frac{\pi}{2})$, condition number $= \dfrac{1.9279 \times 40.86}{-6.314} = -11.2$

For $\tilde{x} = \frac{\pi}{2} + 0.1(\frac{\pi}{2})$, condition number $= \dfrac{1.5865 \times 4053}{64.66} = -101$

Thus the function is ill-conditioned for $\tilde{x} = \frac{\pi}{2} \pm 0.1(\frac{\pi}{2})$

The major cause is the derivative of the function which approaches infinity as *x* approaches $\pi/2$.

Numerical Stability is affected by the number of the significant digits the computer keeps on, if we use a machine that keeps on the first four floating-point digits, a good example on *loss of significance* is given by these two *equivalent* functions

$$f(x) = x\left(\sqrt{x+1} - \sqrt{x}\right) \text{ and } g(x) = \frac{x}{\sqrt{x+1} + \sqrt{x}}.$$

f(x) ≡ g(x)

$$\begin{aligned} f(x) &= x\left(\sqrt{x+1} - \sqrt{x}\right) \\ &= x\left(\sqrt{x+1} - \sqrt{x}\right)\frac{\sqrt{x+1} + \sqrt{x}}{\sqrt{x+1} + \sqrt{x}} \\ &= x\frac{(\sqrt{x+1})^2 - (\sqrt{x})^2}{\sqrt{x+1} + \sqrt{x}} \\ &= x\frac{x+1-x}{\sqrt{x+1} + \sqrt{x}} \\ &= x\frac{1}{\sqrt{x+1} + \sqrt{x}} \\ &= \frac{x}{\sqrt{x+1} + \sqrt{x}} \end{aligned}$$

Subtractive Cancellation

$$\begin{aligned} f(500) &= 500\left(\sqrt{501} - \sqrt{500}\right) \\ &= 500\,(22.3830 - 22.3607) \\ &= 500(0.0223) = \boxed{11.1500} \end{aligned}$$

$$\begin{aligned} g(500) &= \frac{500}{\sqrt{501} + \sqrt{500}} \\ &= \frac{500}{22.3830 + 22.3607} \\ &= \frac{500}{44.7437} = \boxed{11.1748} \end{aligned}$$

The true value for the result is 11.174755..., which is exactly *g*(500) = 11.1748 after rounding the result to 4decimal digits.

Ways to Reduce Numerical Errors

- The total numerical error is the sum of truncation and round-off errors.
- In general the only way to minimize round-off errors is to increase the number of significant figures (use double precision).
- Round-off error will increase due to increase in the number of computations.
- Truncation error associated with discretization can be reduced by reducing the step size. But this will increase the computation steps.
- Since modern computers can carry significant figures (by using double precision, for example), a practical way to reduce numerical error is to decrease step size, at the expense of longer computing time.

Roots of Equations

In some simple equations, roots can be solved analytically

$$f(x) = ax + b = 0 \qquad \Rightarrow \qquad x = -\frac{b}{a}$$

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

But for many other equation, roots may only be solved numerically:

$$f(x) = ax^5 + bx^4 + cx^3 + dx^2 + ex + f = 0 \qquad \Rightarrow \qquad x = ?$$

$$f(x) = \sin x + x = 0 \qquad \Rightarrow \qquad x = ?$$

This section will explore some numerical methods for finding roots of equations.

Methods for Finding Roots of Equations

1. Graphical Method

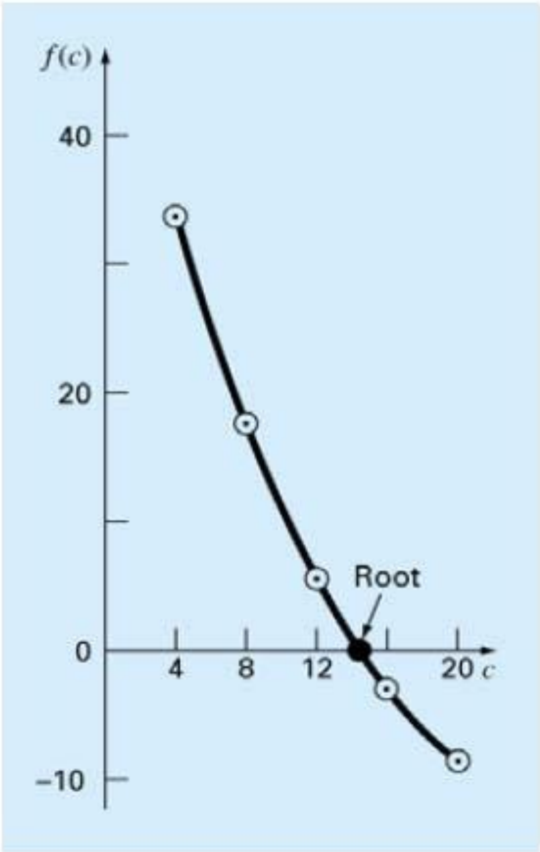
2. Bracketing Methods
   a. Bisection Method
   b. False-Position Method

3. Open Methods
   a. Newton-Raphson Method
   b. Secant Method

Graphical Method (1)

A simple method for getting an estimate of the root of the equation f(x)=0 is to make a plot and observe where it crosses the x axis.

Example: Determine the drag coefficient *c* needed for a parachutist of mass *m* = 68.1 kg to have a velocity of 40 m/sec after free-falling for 10 sec.



The mathematical model is given as follows:

$$v(t) = \frac{gm}{c}(1 - e^{-(c/m)t})$$

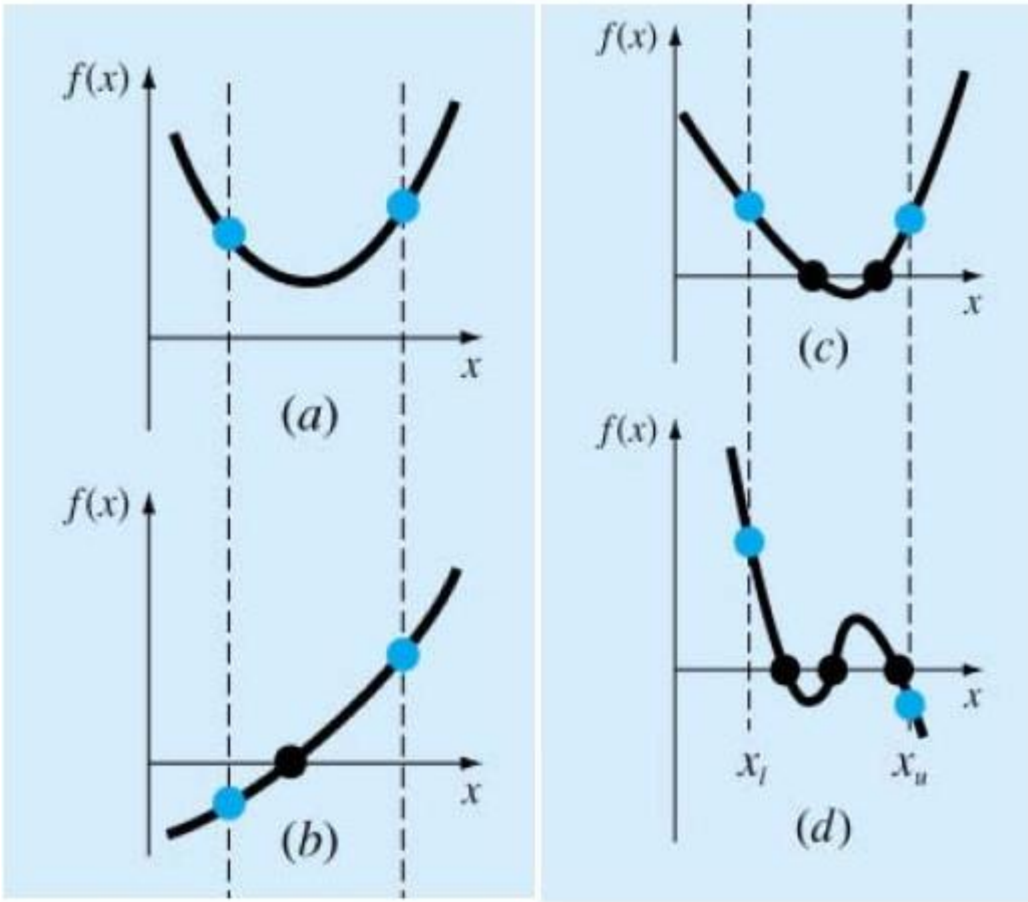
$$40 = \frac{9.8(68.1)}{c}(1 - e^{-(c/68.1)10})$$

$$f(c) = \frac{667.38}{c}(1 - e^{-0.146843c}) - 40 = 0$$

Visual inspection provides a rough estimate of the root of 14.

Graphical Method (2)

Number of roots in an interval

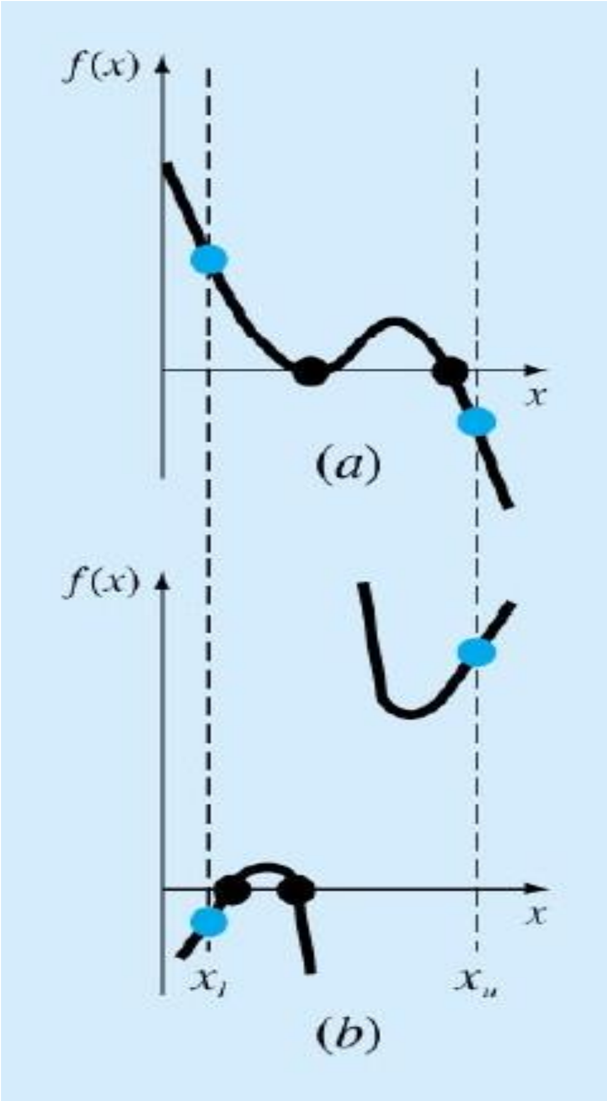


If both $f(x_l)$ and $f(x_u)$ have the *same* sign, either there will be *no* roots or there will be an *even* number of roots within the interval.

If $f(x_l)$ and $f(x_u)$ are of *opposite* signs, there will be an *odd* number of roots within the interval.

Graphical Method (3)

Special Cases:

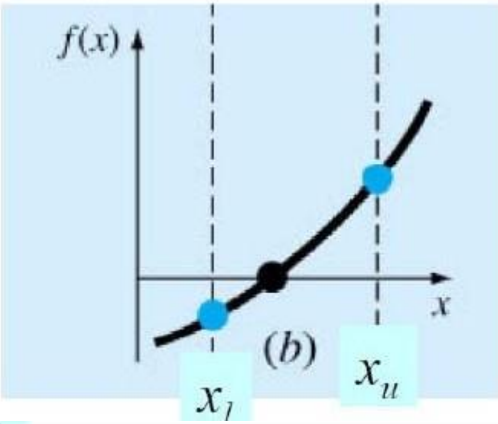


(a) *Multiple* root that occurs when the function is *tangential to the x axis*.

For this case, although the end points are of opposite signs, there are an even number of roots.

(b) Discontinuous function where end points of opposite signs.
There are an even number of roots.

Bisection Method (1)



This method exploits the fact that a function typically *changes sign in the vicinity of a root*. It requires two initial guesses on either side of the root.

$$f(x_l)\, f(x_u) < 0$$

Step 1: Choose lower x_l and upper x_u guesses for the root such that the function changes sign over the interval. This can be checked by ensuring that $f(x_l)f(x_u) < 0$.

Step 2: An estimate of the root x_r is determined by

$$x_r = \frac{x_l + x_u}{2}$$

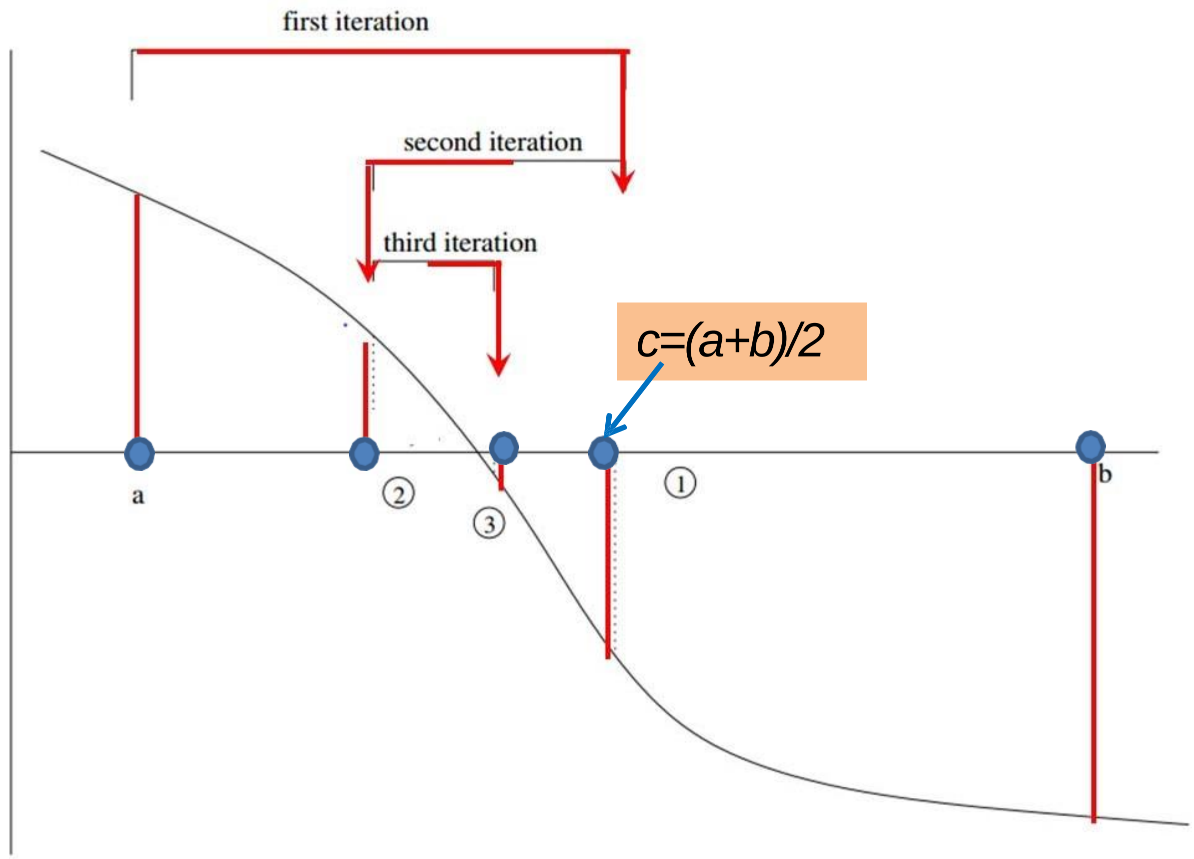
Step 3: Make the following evaluations to determine in which subinterval the root lies:

(a) If $f(x_l)f(x_r) < 0$, the root lies in the lower subinterval. Therefore, set $x_u = x_r$ and return to step 2.

(b) If $f(x_l)f(x_r) > 0$, the root lies in the upper subinterval. Therefore, set $x_l = x_r$ and return to step 2.

(c) If $f(x_l)f(x_r) = 0$, the root equals x_r; terminate the computation.

Bisection Method



Bisection Method (2)

Bisection method is also called binary chopping or interval halving as the interval is always divided into half.

The iterations continue until the relative error, ε_a, *is smaller* than the pre-specified stopping criterion, ε_s.

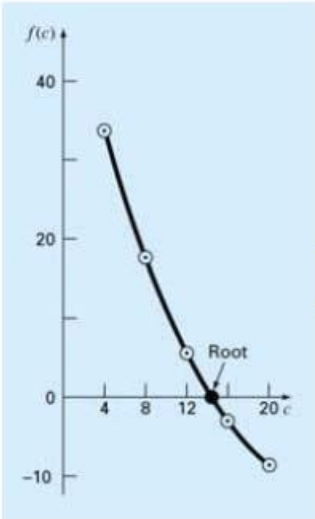
$$\varepsilon_a = \left| \frac{x_r^{new} - x_r^{old}}{x_r^{new}} \right| \times 100\% < \varepsilon_s$$

x_r^{new} = The root for the present iteration

x_r^{old} = The root from the previous iteration

Bisection Method

Example 1



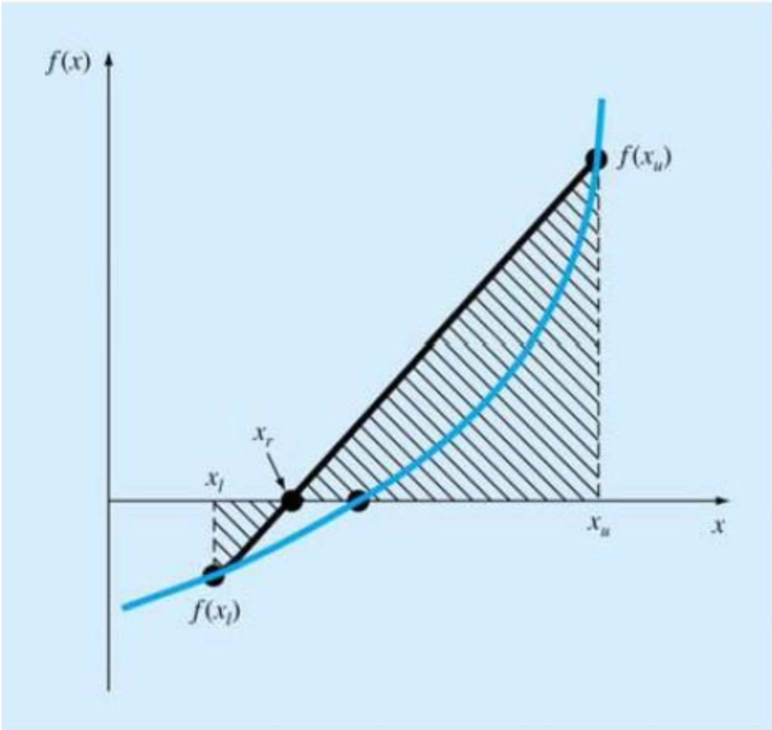
User bisection to find the root correct up to 4 decimal points

$$f(c) = \frac{667.38}{c}(1 - e^{-0.146843c}) - 40 = 0$$

Iteration	c lower	f(c lower)	c upper	f (c upper)	C root	f(c root)	f(c lower)* f(c root)	Approximate error	True error
1	4.00000	34.11490	20.00000	-8.40060	12.00000	6.06690	206.97330		0.18810
2	12.00000	6.06690	20.00000	8.40060	16.00000	2.26880	13.76440	0.25000	0.08253
3	12.00000	6.06690	16.00000	-2.26880	14.00000	1.56870	9.51730	0.14286	0.05278
4	14.00000	1.56870	16.00000	-2.26880	15.00000	-0.42480	-0.66640	0.06667	0.01488
5	14.00000	1.56870	15.00000	-0.42480	14.50000	0.55230	0.86640	0.03448	0.01896
6	14.50000	0.55230	15.00000	-0.42480	14.75000	0.05900	0.03260	0.01695	0.00204
7	14.75000	0.05900	15.00000	-0.42480	14.87500	-0.18410	-0.01090	0.00840	0.00642
8	14.75000	0.05900	14.87500	-0.18410	14.81250	-0.06290	-0.00370	0.00422	0.00219
9	14.75000	0.05900	14.81250	-0.06290	14.78130	-0.00200	-0.00010	0.00211	0.00007
10	14.75000	0.05900	14.78130	-0.00200	14.76560	0.02840	0.00170	0.00106	0.00098
11	14.76560	0.02840	14.78130	-0.00200	14.77340	0.01320	0.00040	0.00053	0.00045
12	14.77340	0.01320	14.78130	-0.00200	14.77730	0.00560	0.00010	0.00026	0.00019
13	14.77730	0.00560	14.78130	-0.00200	14.77930	0.00180	0.00000	0.00013	0.00006
14	14.77930	0.00180	14.78130	-0.00200	14.78030	-0.00010	0.00000	0.00007	0.00001
15	14.77930	0.00180	14.78030	-0.00010	14.77980	0.00080	0.00000	0.00003	0.00003
16	14.77980	0.00080	14.78030	-0.00010	14.78000	0.00030	0.00000	0.00002	0.00001
17	14.78000	0.00030	14.78030	-0.00010	14.78020	0.00010	0.00000	0.00001	0.00000
18	14.78020	0.00010	14.78030	-0.00010	14.78020	0.00000	0.00000		

Root correct up to 4 decimal points

False Position Method (1) - *regula falsi*



A shortcoming of the Bisection method is that, in dividing the interval from x_l to x_u into equal halves, no account is taken of the magnitudes of $f(x_l)$ *and* $f(x_u)$.

An alternative method accounting for the closeness of $f(x_l)$ and $f(x_u)$ to zero is called **Method of False Position or Linear Interpolation method.**

Using similar triangles, the intersection of the straight line with the *x* axis can be estimated as:

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

False Position Method (2)

The value of x_r computed with the False-Position formula then replaces whichever of the two initial guesses, x_l or x_u. The algorithm is identical to the one for Bisection, except Step 2.

Step 1: Choose lower x_l and upper x_u guesses for the root such that the function changes sign over the interval. This can be checked by ensuring that $f(x_l)f(x_u) < 0$.

Step 2: An estimate of the root x_r is determined by

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

Step 3: Make the following evaluations to determine in which subinterval the root lies:

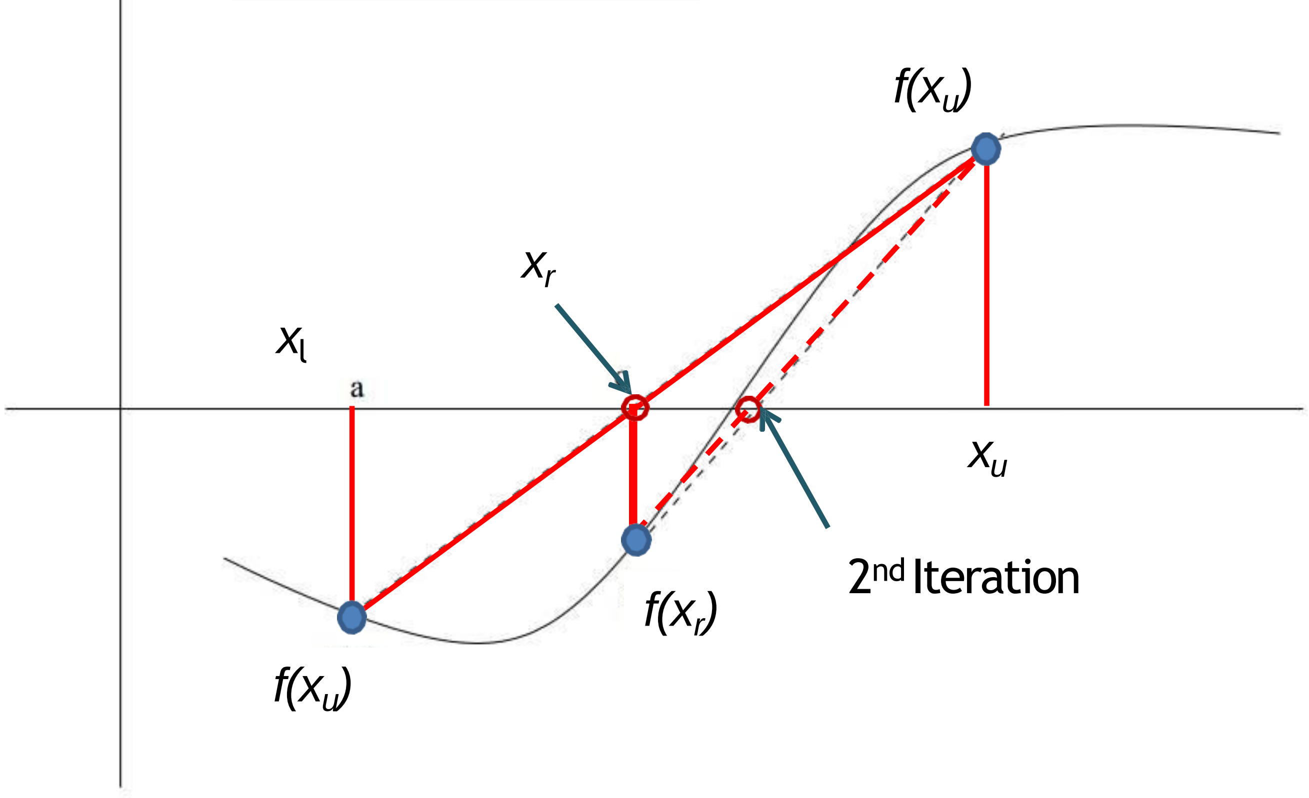
(a) If $f(x_l)f(x_r) < 0$, the root lies in the lower subinterval. Therefore, set $x_u = x_r$ and return to step 2.

(b) If $f(x_l)f(x_r) > 0$, the root lies in the upper subinterval. Therefore, set $x_l = x_r$ and return to step 2.

(c) If $f(x_l)f(x_r) = 0$, the root equals x_r; terminate the computation.

False Position (Interpolation) Method

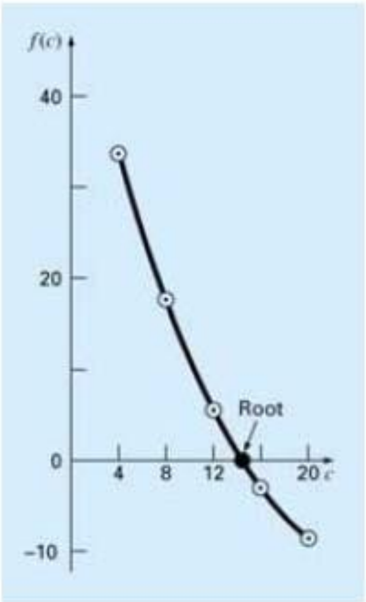
$$x_r = x_u - \frac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)}$$



False Position Method

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

Example 2



Use *False-Position* method to find the root correct up to 4 decimal points.

$$f(c) = \frac{667.38}{c}(1 - e^{-0.146843c}) - 40 = 0$$

It needs less no. of iterations than the Bisection method.

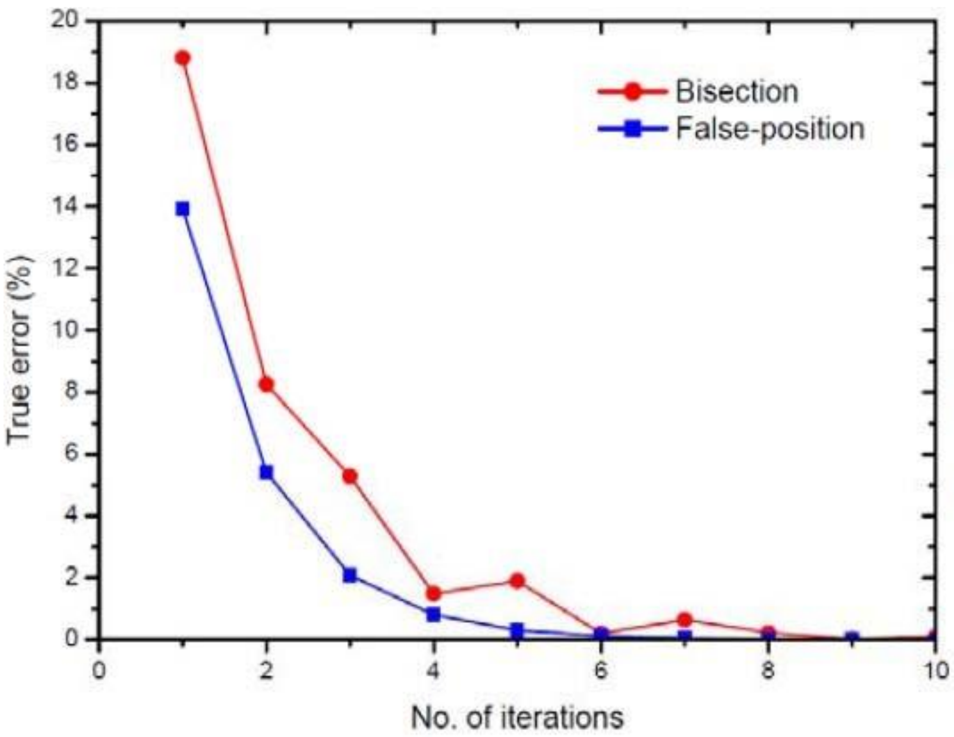
One-sided

Iteration	c lower	f(c lower)	c upper	f (c upper)	c root	f(c root)	f(c lower) *f(c root)	Approximate error	True error
1	4.0000	34.1149	20.0000	-8.4006	16.8386	-3.7096	-126.5536		0.1393
2	4.0000	34.1149	16.8386	-3.7096	15.5794	-1.5106	-51.5331	0.0808	0.0541
3	4.0000	34.1149	15.5794	-1.5106	15.0884	-0.5937	-20.2551	0.0325	0.0209
4	4.0000	34.1149	15.0884	-0.5937	14.8988	-0.2301	-7.8482	0.0127	0.0080
5	4.0000	34.1149	14.8988	-0.2301	14.8258	-0.0886	-3.0240	0.0049	0.0031
6	4.0000	34.1149	14.8258	-0.0886	14.7977	-0.0341	-1.1626	0.0019	0.0012
7	4.0000	34.1149	14.7977	-0.0341	14.7869	-0.0131	-0.4466	0.0007	0.0005
8	4.0000	34.1149	14.7869	-0.0131	14.7828	-0.0050	-0.1715	0.0003	0.0002
9	4.0000	34.1149	14.7828	-0.0050	14.7812	-0.0019	-0.0659	0.0001	0.0001
10	4.0000	34.1149	14.7812	-0.0019	14.7806	-0.0007	-0.0253	0.0000	0.0000
11	4.0000	34.1149	14.7806	-0.0007	14.7804	-0.0003	-0.0097	0.0000	0.0000
12	4.0000	34.1149	14.7804	-0.0003	14.7803	-0.0001	-0.0037	0.0000	0.0000
13	4.0000	34.1149	14.7803	-0.0001	14.7802	0.0000	-0.0014	0.0000	0.0000
14	4.0000	34.1149	14.7802	0.0000	14.7802	0.0000	-0.0005		

Correct to 4 decimal places →

False Position Method

Example 2 (cont'd)



The error for False-Position method decreases much faster than for Bisection method because of the more efficient scheme for root location in the former.

False Position Method

Example 3

Use Bisection and False-Position methods to locate the root of

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

between $x = 0$ and 1.3.

Solution:

The exact root is $x = 1.0$.

Method of Bisection

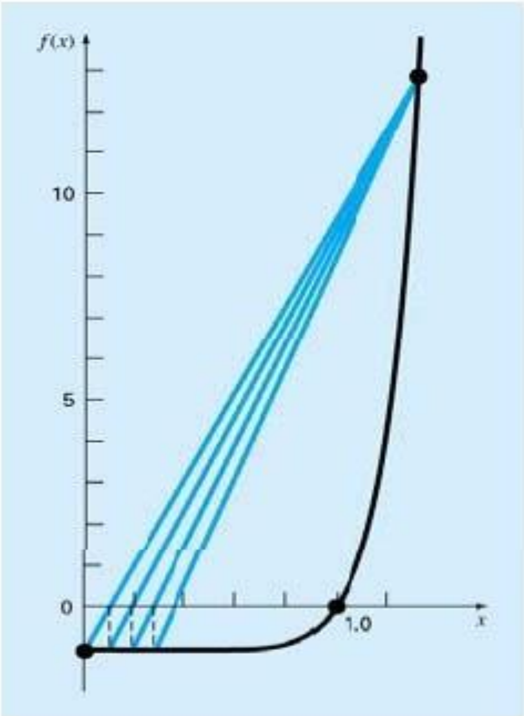
Iteration	x lower	f(x lower)	x upper	f (x upper)	x root	f(x root)	f(x lower) *f(x root)	Approximate error	True error
1	0.0000	-1.0000	1.3000	12.7858	0.6500	-0.9865	0.9865		35.00%
2	0.6500	-0.9865	1.3000	12.7858	0.9750	-0.2237	0.2207	33.30%	2.50%
3	0.9750	-0.2237	1.3000	12.7858	1.1375	2.6267	-0.5875	14.30%	13.80%
4	0.9750	-0.2237	1.1375	2.6267	1.0563	0.7285	-0.1629	7.70%	5.60%
5	0.9750	-0.2237	1.0563	0.7285	1.0156	0.1677	-0.0375	4.00%	1.60%

After 5 iterations, the true error is reduced to less than 2%.

Example 3 (cont'd)

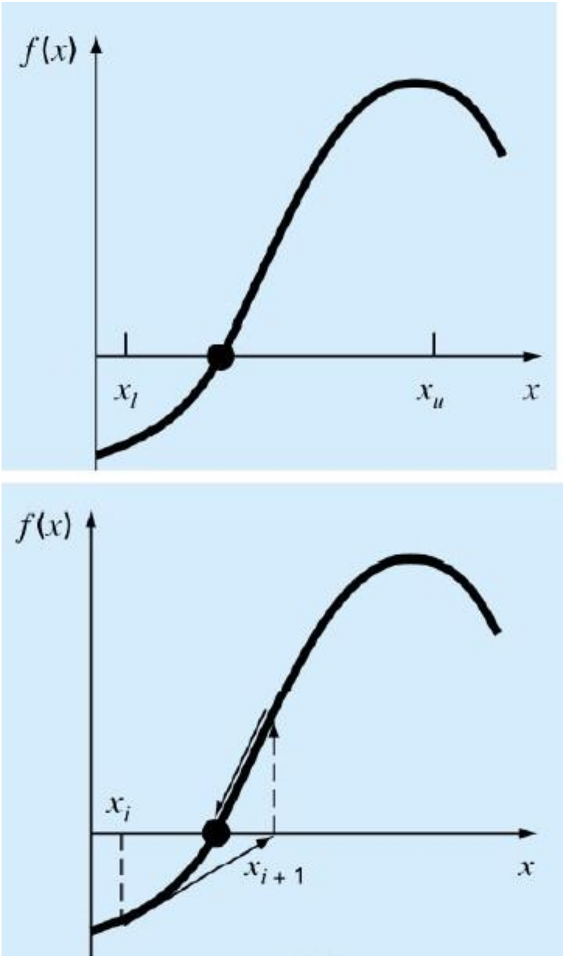
Method of False-Position

Iteration	x lower	f(x lower)	x upper	f (x upper)	x root	f(x root)	f(x lower) *f(x root)	Approximate error	True error
1	0.0000	-1.0000	1.3000	12.7858	0.0943	-1.0000	1.0000		90.60%
2	0.0943	-1.0000	1.3000	12.7858	0.1818	-1.0000	1.0000	48.10%	81.80%
3	0.1818	-1.0000	1.3000	12.7858	0.2629	-1.0000	1.0000	30.90%	73.70%
4	0.2629	-1.0000	1.3000	12.7858	0.3381	-1.0000	1.0000	22.30%	66.20%
5	0.3381	-1.0000	1.3000	12.7858	0.4079	-0.9999	0.9999	17.10%	59.20%



After 5 iterations, the true error has only been reduced to about 59%.
The curve violates the premise upon which False-Position was based – that is
$f(x_l)$ is much closer to zero than $f(x_u)$, then the root is closer to x_l than x_u. This illustrates a major weakness of the False-Position method: its one-sidedness, which may lead to poor convergence.

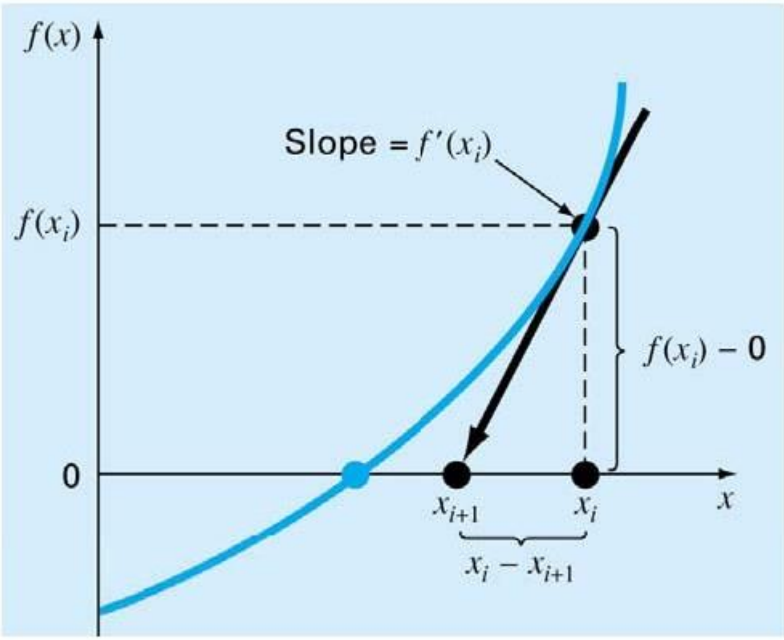
Bracketing vs Open Method



The Bisection and False-Position methods are categorised as **bracketing methods,** where the root is located within an interval prescribed by a lower and an upper bound. Repeated application of these methods always results in closer estimates of the true value of the root. Such methods are said to be **convergent.**

In contrast, the **open methods** require only a single starting value of *x*.

The Newton-Raphson Method



The Newton-Raphson method is one of the most widely used open methods.

1. Take an initial guess x_i.

2. Draw a tangent from the point $[x_i, f(x_i)]$.

3. The point where the tangent crosses the x - axis represents an improved estimate of the root.

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

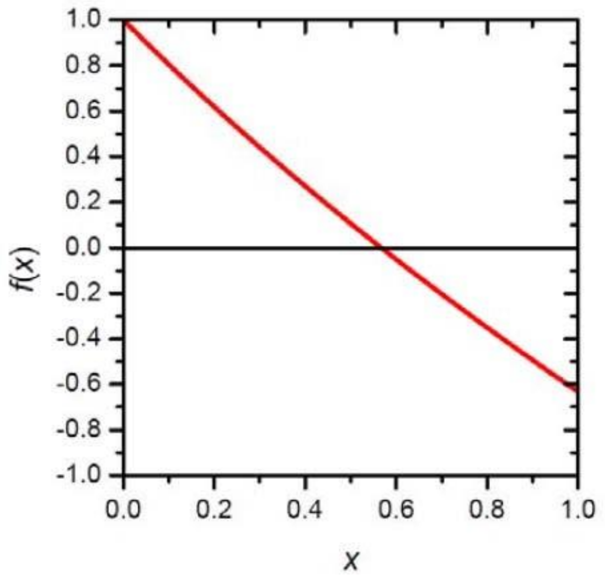
$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Newton-Raphson formula

The Newton-Raphson Method

Example 4

Use the Newton-Raphson method to estimate the root of $f(x) = e^{-x} - x = 0$



Solution:

$$f'(x) = -e^{-x} - 1$$

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

Take an initial guess of $x_0 = 0$.

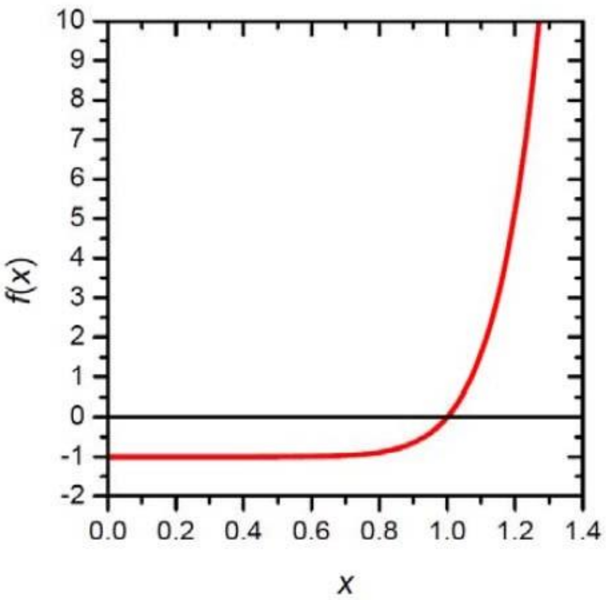
i	x_i	true error
0	0	
1	0.5000000000	11.84%
2	0.5663110032	0.15%
3	0.5671431650	0.00%
4	0.5671432904	0.00%
5	0.5671432904	0.00%

Take an initial guess of $x_0 = 20$.

i	x_i	true error
0	20	
1	0.0000000433	100.00%
2	0.5000000108	11.84%
3	0.5663110035	0.15%
4	0.5671431650	0.00%
5	0.5671432904	0.00%
6	0.5671432904	0.00%

Example 5

Using the Newton-Raphson method, determine the root of $f(x) = x^{10} - 1 = 0$



Solution: $f'(x) = 10x^9$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} = x_i - \frac{x_i^{10} - 1}{10x_i^9}$$

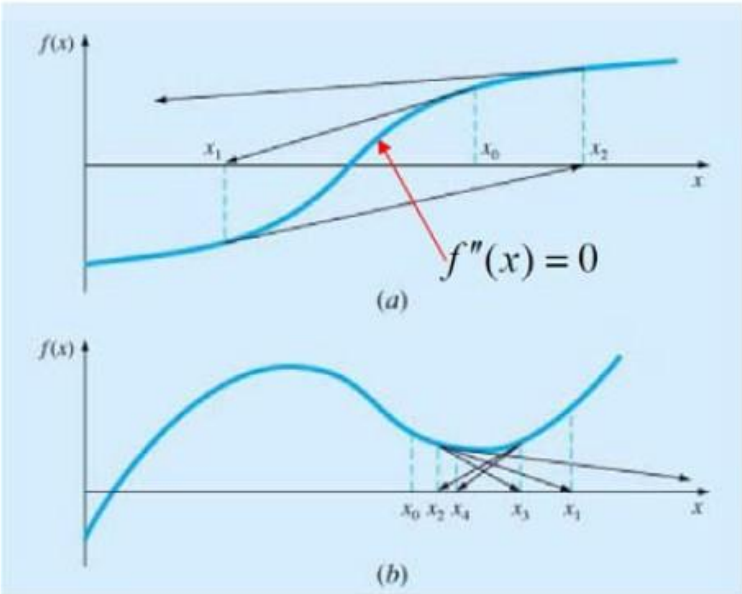
i	X_i
0	0.1
1	100000000
2	90000000
3	81000000
4	72900000
5	65610000
6	59049000
7	53144100
8	47829690
9	43046721
10	38742049
170	2
171	2
172	1.499416
173	1.352085
174	1.223498
175	1.117425
176	1.042498
177	1.007006
178	1.000215
179	1.000000

i	X_i
0	0.700000
1	3.108093
2	2.797288
3	2.517568
4	2.265836
5	2.039316
6	1.835548
7	1.652416
8	1.488263
9	1.342229
10	1.215078
11	1.110891
12	1.038613
13	1.005859
14	1.000151
15	1.000000

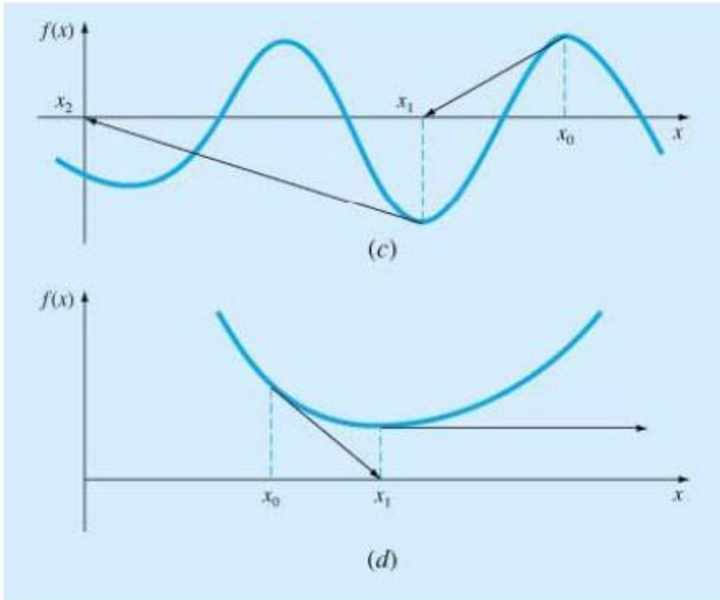
i	X_i
0	2.000000
1	1.800195
2	1.620679
3	1.459908
4	1.317237
5	1.193889
6	1.094792
7	1.029573
8	1.003544
9	1.000056
10	1.000000

Pitfalls of the Newton-Raphson Method

(a) an inflection point occurs
in the vicinity of the root



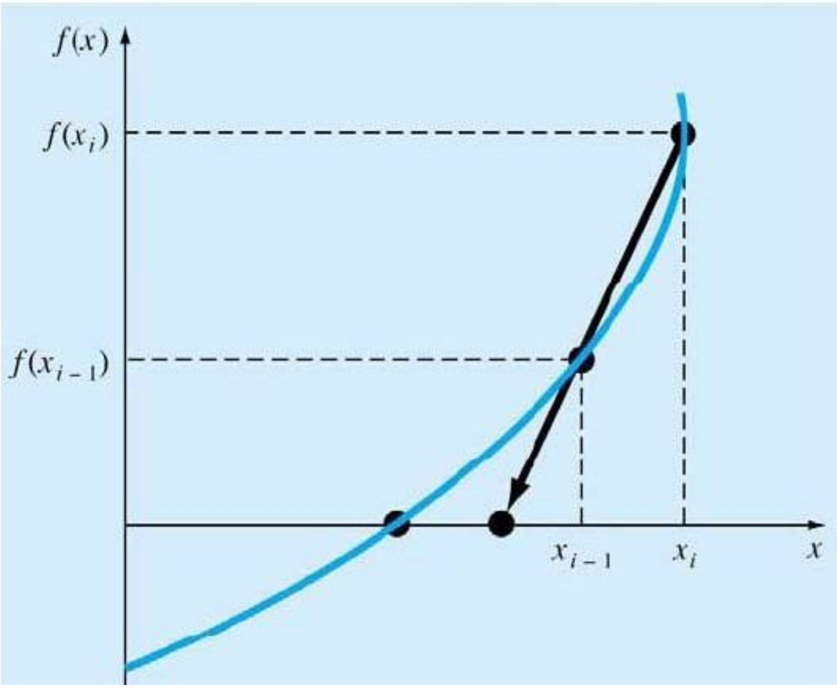
(b) oscillate around a local maximum or minimum



(d) near-zero slopes

The Secant Method

The potential problem in implementing the Newton-Raphson method is the *evaluation of the derivative*. There are certain functions whose derivatives may be difficult to evaluate. For these cases, the derivative can be *approximated by a backward finite divided difference*.



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

Substitute this to the Newton-Raphson formula:

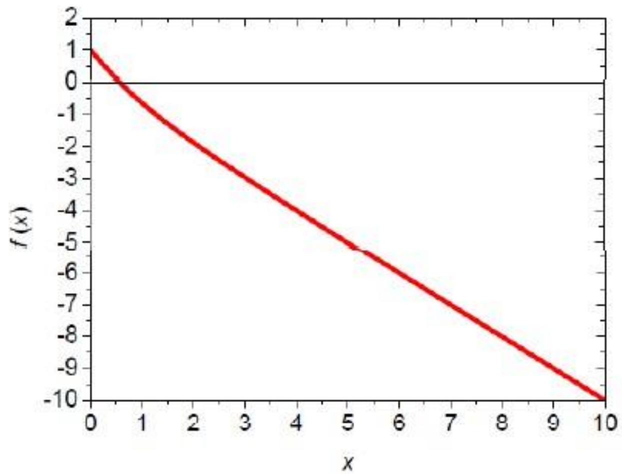
$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

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

Example 6

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

Use the secant method to estimate the root of $f(x) = e^{-x} - x = 0$



The final results do not depend on the initial estimates.

Solution:

Start with initial estimates of $x_{-1} = 0$ and $x_0 = 0.1$.

i	X_{i-1}	$f(X_{i-1})$	X_i	$f(X_i)$	X_{i+1}	True Error
0	0.000000	1.000000	0.100000	0.804837	0.512393	9.65%
1	0.100000	0.804837	0.512393	0.086667	0.562160	0.88%
2	0.512393	0.086667	0.562160	0.007817	0.567093	0.01%
3	0.562160	0.007817	0.567093	0.000078	0.567143	0.00%
4	0.567093	0.000078	0.567143	0.000000	0.567143	0.00%

Start with initial estimates of $x_{-1} = 0$ and $x_0 = 1$.

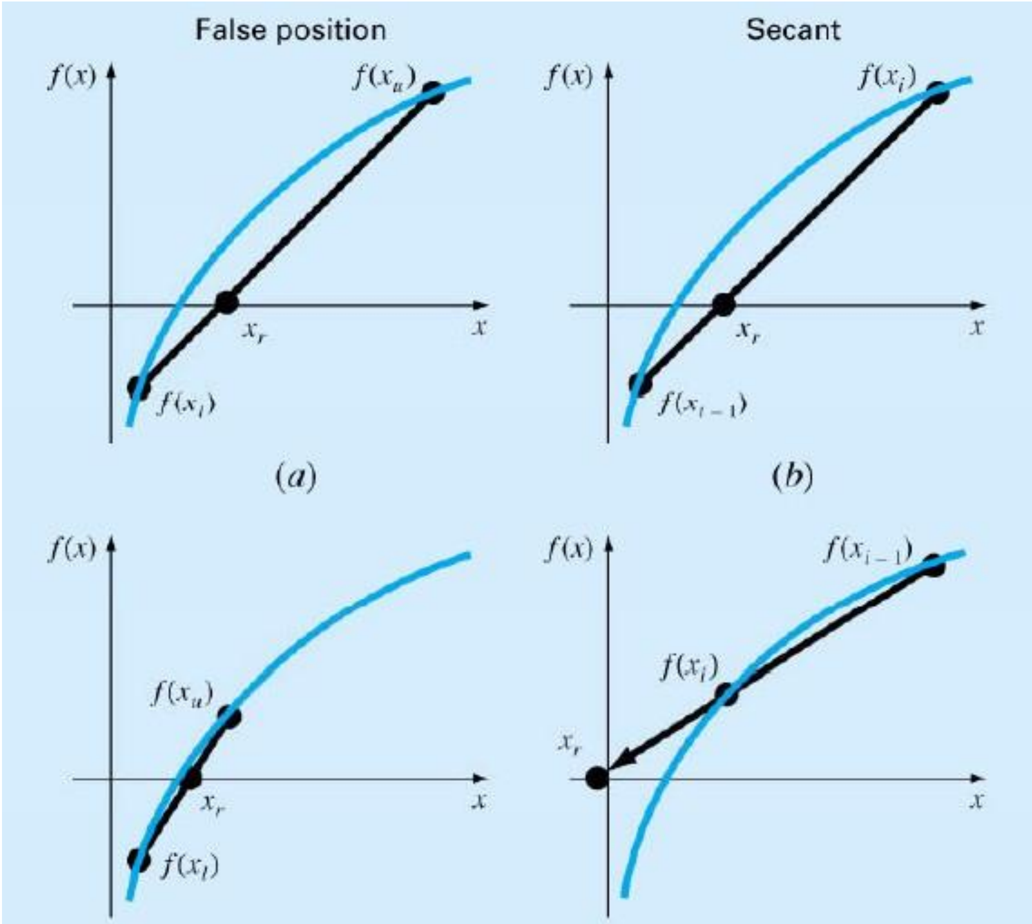
i	X_{i-1}	$f(X_{i-1})$	X_i	$f(X_i)$	X_{i+1}	True Error
0	0.000000	1.000000	1.000000	-0.632121	0.612700	8.03%
1	1.000000	-0.632121	0.612700	-0.070814	0.563838	0.58%
2	0.612700	-0.070814	0.563838	0.005182	0.567170	0.00%
3	0.563838	0.005182	0.567170	-0.000042	0.567143	0.00%
4	0.567170	-0.000042	0.567143	0.000000	0.567143	0.00%

Start with initial estimates of $x_{-1} = 10$ and $x_0 = 5$.

i	X_{i-1}	$f(X_{i-1})$	X_i	$f(X_i)$	X_{i+1}	True Error
0	10.000000	-9.999955	5.000000	-4.993262	0.013413	97.64%
1	5.000000	-4.993262	0.013413	0.973264	0.826829	45.79%
2	0.013413	0.973264	0.826829	-0.389394	0.594386	4.80%
3	0.826829	-0.389394	0.594386	-0.042485	0.565920	0.22%
4	0.594386	-0.042485	0.565920	0.001918	0.567149	0.00%
5	0.565920	0.001918	0.567149	-0.000009	0.567143	0.00%
6	0.567149	-0.000009	0.567143	0.000000	0.567143	0.00%

Example 7

Use the false-position and secant methods to estimate the root of $f(x) = \ln x = 0$. Start the computation with x values of 0.5 and 5.



Example 7

False-Position method

Iteration	x lower	f(x lower)	x upper	f (x upper)	x root	f(x root)	f(x lower) *f(x root)	Approximate error	True error
1	0.5000	-0.6931	5.0000	1.6094	1.8546	0.6177	-0.4281		85.50%
2	0.5000	-0.6931	1.8546	0.6177	1.2163	0.1958	-0.1357	52.50%	21.60%
3	0.5000	-0.6931	1.2163	0.1958	1.0585	0.0569	-0.0394	14.90%	5.90%
4	0.5000	-0.6931	1.0585	0.0569	1.0162	0.0160	-0.0111	4.20%	1.60%
5	0.5000	-0.6931	1.0162	0.0160	1.0045	0.0045	-0.0031	1.20%	0.40%
6	0.5000	-0.6931	1.0045	0.0045	1.0013	0.0013	-0.0009	0.30%	0.10%
7	0.5000	-0.6931	1.0013	0.0013	1.0003	0.0003	-0.0002	0.10%	0.00%
8	0.5000	-0.6931	1.0003	0.0003	1.0001	0.0001	-0.0001	0.00%	0.00%
9	0.5000	-0.6931	1.0001	0.0001	1.0000	0.0000	0.0000	0.00%	0.00%
10	0.5000	-0.6931	1.0000	0.0000	1.0000	0.0000	0.0000	0.00%	0.00%

Secant method

i	X_{i-1}	$f(X_{i-1})$	X_i	$f(X_i)$	X_{i+1}
0	0.5000	-0.6931	5.0000	1.6094	1.8546
1	5.0000	1.6094	1.8546	0.6177	-0.1044
2	1.8546	0.6177	-0.1044	#NUM!	#NUM!

i	X_{i-1}	$f(X_{i-1})$	X_i	$f(X_i)$	X_{i+1}	True error
0	5.0000	1.6094	0.5000	-0.6931	1.8546	85.46%
1	0.5000	-0.6931	1.8546	0.6177	1.2163	21.63%
2	1.8546	0.6177	1.2163	0.1958	0.9200	8.00%
3	1.2163	0.1958	0.9200	-0.0834	1.0085	0.85%
4	0.9200	-0.0834	1.0085	0.0085	1.0003	0.03%
5	1.0085	0.0085	1.0003	0.0003	1.0000	0.00%
6	1.0003	0.0003	1.0000	0.0000	1.0000	0.00%

Although the secant method may be divergent, when it converges it usually does so at a quicker rate than the False-Position method.

The Modified Secant Method

Rather than using two arbitrary values to estimate the derivative, an alternative approach involves a fractional perturbation of the independent variable to estimate the derivative.

$$f'(x) \cong \frac{f(x_i + \delta x_i) - f(x_i)}{\delta x_i}$$

Substitute this to the Newton-Raphson formula:

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

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

Example 8

Use the modified secant method to estimate the root of $f(x) = e^{-x} - x = 0$

Solution:

delta = 0.5

i	X_i	$f(X_i)$	$X_i+\delta X_i$	$f(X_i+\delta X_i)$	X_{i+1}	True error
1	1.00000000	-0.63212056	1.50000000	-1.27686984	0.50979351	10.11%
2	0.50979351	0.09082607	0.76469027	-0.29921219	0.56914992	0.35%
3	0.56914992	-0.00314354	0.85372489	-0.42789907	0.56704383	0.02%
4	0.56704383	0.00015586	0.85056575	-0.42339256	0.56714817	0.00%
5	0.56714817	-0.00000765	0.85072225	-0.42361591	0.56714305	0.00%
6	0.56714305	0.00000038	0.85071458	-0.42360496	0.56714330	0.00%
7	0.56714330	-0.00000002	0.85071495	-0.42360549	0.56714329	0.00%
8	0.56714329	0.00000000	0.85071493	-0.42360547	0.56714329	0.00%

delta = 0.01

i	X_i	$f(X_i)$	$X_i+\delta X_i$	$f(X_i+\delta X_i)$	X_{i+1}	True error
1	1.00000000	-0.63212056	1.01000000	-0.64578102	0.53726267	5.27%
2	0.53726267	0.04708295	0.54263529	0.03857927	0.56700969	0.02%
3	0.56700969	0.00020938	0.57267978	-0.00866780	0.56714342	0.00%
4	0.56714342	-0.00000021	0.57281486	-0.00887906	0.56714329	0.00%
5	0.56714329	0.00000000	0.57281472	-0.00887884	0.56714329	0.00%

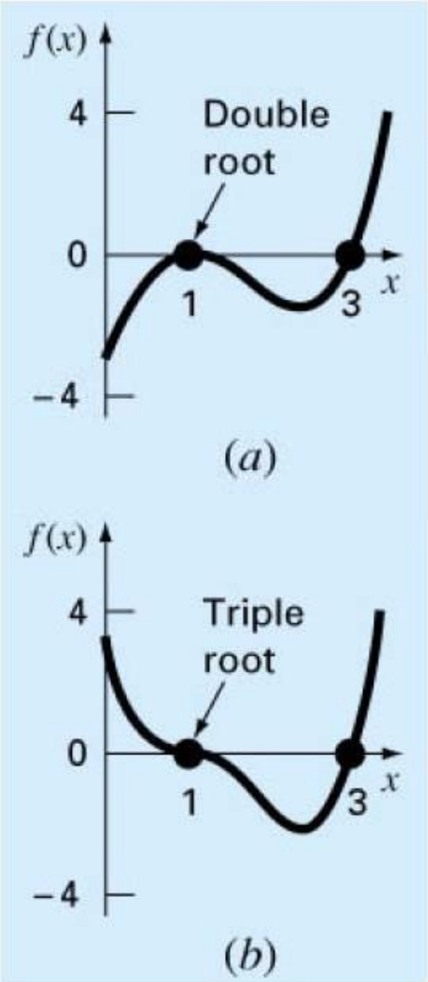
delta = 0.0001

i	X_i	$f(X_i)$	$X_i+\delta X_i$	$f(X_i+\delta X_i)$	X_{i+1}	True error
1	1.00000000	-0.63212056	1.00010000	-0.63225734	0.53787663	5.16%
2	0.53787663	0.04611033	0.53793042	0.04602513	0.56698721	0.03%
3	0.56698721	0.00024460	0.56704391	0.00015574	0.56714329	0.00%
4	0.56714329	0.00000000	0.56720000	-0.00008887	0.56714329	0.00%

delta = 1.00E-18

i	X_i	$f(X_i)$	$X_i+\delta X_i$	$f(X_i+\delta X_i)$	X_{i+1}	True error
1	1.00000000	-0.63212056	1.00000000	-0.63212056	#DIV/0!	#DIV/0!
2	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!

Multiple Roots (1)



$f(x) = (x - 1)(x - 1)(x - 3) = x^3 - 5x^2 + 7x - 3$

This function has a ***double root*** because one value of *x* (*x = 1*) makes two terms equal to zero. Graphically, this corresponds to the curve touching the *x* axis tangentially at the double root.

$f(x) = (x-1)(x-1)(x-1)(x-3) = x^4 - 6x^3 + 12x^2 - 10x + 3$

A ***triple root*** corresponds to the case where one *x* value makes three terms in an equation equal to zero. Note that the function is tangent to the axis at the root.

In general, a ***multiple root*** corresponds to a point where a function is tangent to the *x axis.*

Multiple Roots (2)

Multiple roots pose some difficulties:

1. The fact that the function may not change sign at a multiple root precludes the use of the bracketing methods.

2. The fact that not only *f*(*x*) but also *f* ´(*x*) goes to zero at the root poses problems for both the Newton-Raphson and secant methods. This could result in division by zero when the solution converges very close to the root.

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

To avoid this problem, Ralston and Robinowitz (1978) suggested the ***modified Newton-Raphson*** method. A new function u(x) is defined as:

$$u(x) = \frac{f(x)}{f'(x)}$$

Multiple Roots (3)

Ralston and Robinowitz (1978) showed that *u*(*x*) has roots at all the same locations as the original function *f*(*x*).

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

Knowing that $$u'(x) = \frac{f'(x)f'(x) - f(x)f''(x)}{[f'(x)]^2}$$

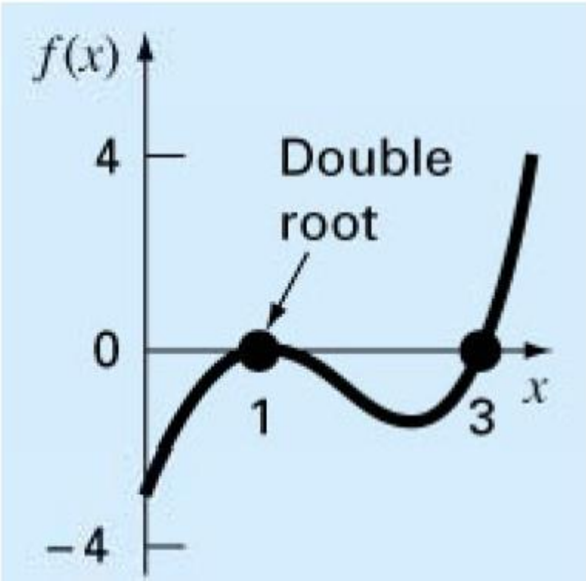
$$x_{i+1} = x_i - \frac{f(x_i)f'(x_i)}{[f'(x_i)]^2 - f(x_i)f''(x_i)}$$

Modified Newton-Raphson

Example 9

Use the standard and modified Newton-Raphson methods to evaluate the roots of the following equation.

$$f(x) = (x - 3)(x - 1)(x - 1) = x^3 - 5x^2 + 7x - 3$$



<u>Solution</u>:

Standard Newton-Raphson method:

$$f'(x) = 3x^2 - 10x + 7$$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} = x_i - \frac{x_i^3 - 5x_i^2 + 7x_i - 3}{3x_i^2 - 10x_i + 7}$$

Example 9 (cont'd)

Standard Newton-Raphson method

i	x_i	True error
0	0.000000000	100.00%
1	0.428571400	57.14%
2	0.685714300	31.43%
3	0.832865400	16.71%
4	0.913329900	8.67%
5	0.955783300	4.42%
6	0.977655100	2.23%
7	0.988766200	1.12%
8	0.994367400	0.56%
9	0.997179800	0.28%
10	0.998588900	0.14%
11	0.999294200	0.07%
12	0.999647000	0.04%
13	0.999823500	0.02%
14	0.999911700	0.01%
15	0.999955900	0.00%
16	0.999977900	0.00%
17	0.999989000	0.00%
18	0.999994500	0.00%
19	0.999997200	0.00%
20	0.999998600	0.00%
21	0.999999300	0.00%
22	0.999999700	0.00%
23	0.999999800	0.00%
24	0.999999900	0.00%
25	1.000000000	0.00%
26	1.000000000	0.00%

i	x_i	True error
0	1.5000000	50.00%
1	1.2000000	20.00%
2	1.0941176	9.41%
3	1.0458675	4.59%
4	1.0226614	2.27%
5	1.0112654	1.13%
6	1.0056167	0.56%
7	1.0028044	0.28%
8	1.0014012	0.14%
9	1.0007004	0.07%
10	1.0003501	0.04%
11	1.0001750	0.02%
12	1.0000875	0.01%
13	1.0000438	0.00%
14	1.0000219	0.00%
15	1.0000109	0.00%
16	1.0000055	0.00%
17	1.0000027	0.00%
18	1.0000014	0.00%
19	1.0000007	0.00%
20	1.0000003	0.00%
21	1.0000002	0.00%
22	1.0000001	0.00%
23	1.0000000	0.00%
24	1.0000000	0.00%

i	x_i	True error
0	2.5000000	16.67%
1	4.0000000	33.33%
2	3.4000000	13.33%
3	3.1000000	3.33%
4	3.0086957	0.29%
5	3.0000746	0.00%
6	3.0000000	0.00%
7	3.0000000	0.00%

i	x_i	True error
0	10.0000000	233.33%
1	7.2608696	142.03%
2	5.4562660	81.88%
3	4.2879438	42.93%
4	3.5657732	18.86%
5	3.1731521	5.77%
6	3.0238001	0.79%
7	3.0005469	0.02%
8	3.0000003	0.00%
9	3.0000000	0.00%
10	3.0000000	0.00%

Example 9 (cont'd)

Modified Newton-Raphson method:

$f'(x) = 3x^2 - 10x + 7$

$f''(x) = 6x - 10$

$$x_{i+1} = x_i - \frac{f(x_i)f'(x_i)}{[f'(x_i)]^2 - f(x_i)f''(x_i)} = x_i - \frac{(x_i^3 - 5x_i^2 + 7x_i - 3)(3x_i^2 - 10x_i + 7)}{(3x_i^2 - 10x_i + 7)^2 - (x_i^3 - 5x_i^2 + 7x_i - 3)(6x_i - 10)}$$

i	x_i	True error
0	0.0000000	100.00%
1	1.1052632	10.53%
2	1.0030817	0.31%
3	1.0000024	0.00%
4	1.0000000	0.00%
5	1.0000000	0.00%

i	x_i	True error
0	2.5000000	16.67%
1	2.6363636	12.12%
2	2.8202247	5.99%
3	2.9617282	1.28%
4	2.9984787	0.05%
5	2.9999977	0.00%
6	3.0000000	0.00%
7	3.0000000	0.00%

i	x_i	True error
0	5.0000000	66.67%
1	2.3333333	22.22%
2	2.3333333	22.22%
3	2.3333333	22.22%
4	2.3333333	22.22%
5	2.3333333	22.22%
6	2.3333333	22.22%

i	x_i
0	10.0000000
1	1.9050279
2	1.5092165
3	1.1102432
4	1.0033975
5	1.0000029
6	1.0000000
7	1.0000000

i	x_i	True error
0	1.5000000	50.00%
1	1.1052632	10.53%
2	1.0030817	0.31%
3	1.0000024	0.00%
4	1.0000000	0.00%
5	1.0000000	0.00%

System of Nonlinear Equations (1)

To this point, we have focused on the determination of roots of a single equation. A related problem is to locate the roots of a set of simultaneous equations.

$$f_1(x_1, x_2, \cdots, x_n) = 0$$
$$f_2(x_1, x_2, \cdots, x_n) = 0$$
$$\vdots$$
$$f_n(x_1, x_2, \cdots, x_n) = 0$$

In the first part of this course, you have learned how to solve a set of simultaneous ***linear*** equations.

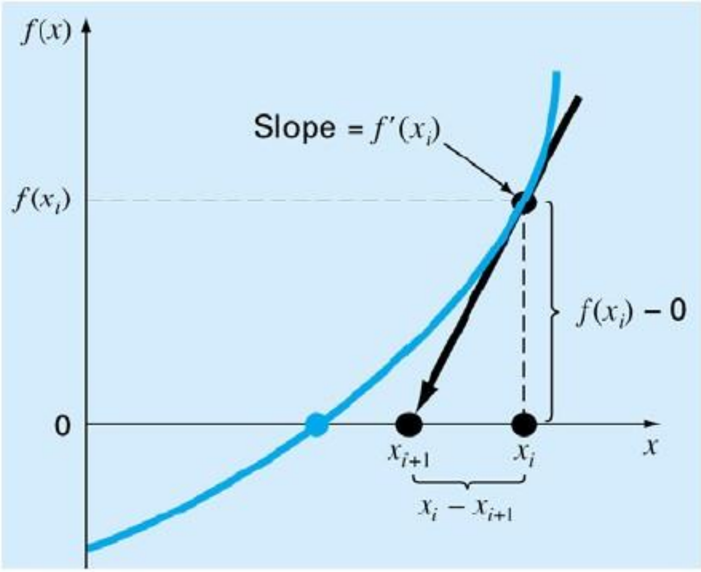
$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n + c_1 = 0$$
$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n + c_2 = 0$$
$$\vdots$$
$$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n + c_n = 0$$

System of Nonlinear Equations (2)

Here, we will use the Newton-Raphson method to solve a set of ***nonlinear*** equations, such as

$$u(x,y) = x^2 + xy - 10 = 0$$

$$v(x,y) = y + 3xy^2 - 57 = 0$$



Recall the Newton-Raphson method for one independent variable:

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

x_i = initial guess

x_{i+1} = the point at which the slope intersects with the *x axis*

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

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

System of Nonlinear Equations (3)

For a set of equations with two independent variables:

$$u(x, y) = 0$$

$$v(x, y) = 0$$

$$u_{i+1} = u_i + (x_{i+1} - x_i)\frac{\partial u_i}{\partial x} + (y_{i+1} - y_i)\frac{\partial u_i}{\partial y}$$

$$v_{i+1} = v_i + (x_{i+1} - x_i)\frac{\partial v_i}{\partial x} + (y_{i+1} - y_i)\frac{\partial v_i}{\partial y}$$

The root estimate corresponds to the values of *x* and *y*, where u_{i+1} and v_{i+1} equal zero.

$$\frac{\partial u_i}{\partial x}x_{i+1} + \frac{\partial u_i}{\partial y}y_{i+1} = -u_i + \frac{\partial u_i}{\partial x}x_i + \frac{\partial u_i}{\partial y}y_i$$

$$\frac{\partial v_i}{\partial x}x_{i+1} + \frac{\partial v_i}{\partial y}y_{i+1} = -v_i + \frac{\partial v_i}{\partial x}x_i + \frac{\partial v_i}{\partial y}y_i$$

System of Nonlinear Equations (4)

Solve these two linear equation simultaneously:

$$x_{i+1} = x_i - \frac{u_i \frac{\partial v_i}{\partial y} - v_i \frac{\partial u_i}{\partial y}}{\frac{\partial u_i}{\partial x}\frac{\partial v_i}{\partial y} - \frac{\partial u_i}{\partial y}\frac{\partial v_i}{\partial x}}$$

$$y_{i+1} = y_i - \frac{v_i \frac{\partial u_i}{\partial x} - u_i \frac{\partial v_i}{\partial x}}{\frac{\partial u_i}{\partial x}\frac{\partial v_i}{\partial y} - \frac{\partial u_i}{\partial y}\frac{\partial v_i}{\partial x}}$$

The denominator of each of these equations is referred to as the determinant of the ***Jacobian*** of the system.

Example 10

Use the multiple-equation Newton-Raphson method to determine the roots of the following equations:

$$u(x, y) = x^2 + xy - 10 = 0$$

$$v(x, y) = y + 3xy^2 - 57 = 0$$

Solution

$$\frac{\partial u}{\partial x} = 2x + y \qquad \frac{\partial u}{\partial y} = x \qquad \frac{\partial v}{\partial x} = 3y^2 \qquad \frac{\partial v}{\partial y} = 1 + 6xy$$

i	x_i	y_i	u_i	v_i	du/dx	du/dy	dv/dx	dv/dy	x_{i+1}	y_{i+x}
0	1.0000	1.0000	-8.0000	-53.0000	3.0000	1.0000	3.0000	7.0000	1.1667	8.5000
1	1.1667	8.5000	1.2778	204.3750	10.8333	1.1667	216.7500	60.5000	1.5670	3.6878
2	1.5670	3.6878	-1.7660	10.6195	6.8217	1.5670	40.8000	35.6718	2.0108	2.8824
3	2.0108	2.8824	-0.1605	-3.9973	6.9041	2.0108	24.9251	35.7765	1.9992	3.0023
4	1.9992	3.0023	-0.0013	0.0620	7.0006	1.9992	27.0411	37.0124	2.0000	3.0000
5	2.0000	3.0000	0.0000	0.0000	7.0000	2.0000	27.0000	37.0000	2.0000	3.0000

Further Readings:

Numerical Methods for Engineers by SC Chapra and RPCanale, 2010 Sixth Edition, McGraw-Hill International Edition.

Chapters 5 & 6.