Root finding Methods

Introduction

A problem of great importance in applied mathematics and engineering is that of determining the roots of an equation of the form

$$f(x) = 0 \tag{1.1}$$

where x and $f(x)$ may be real, complex or vector values.

Equation (1.1) may belong to one of the following types of equations:

(i) Polynomial equations
(ii) Algebraic equations
(iii) Transcendental equations

1 Polynomial Equations

An expression of the form $f(x) = a_0x^n + a_1x^{n-1} + a_2\, x^{n-2} + \ldots + a_n$

where n is a positive integer and $a_0, a_1, a_2, \ldots\ a_n$ are all constants, called a polynomial in x of the nth degree, if $a_0 \neq 0$.

2 Algebraic Equations

An equation of the type, $y = f(x)$, is said to be algebraic if it can be expressed in the form

$$f_0 + f_1 y_1 + f_2 y_2 + \ldots + f_n y_n = 0$$

where f_k is a kth degree polynomial in x.

EXAMPLE

(i) $5x + 3y - 88 = 0$ (linear)

(ii) $101y - 10x + 55xy = 0$ (nonlinear)

3 Transcendental Equations

Any nonalgebraic equations is called a transcendental equation, i.e. it contains some other functions, such as trigonometric, logarithmic, exponential etc.

EXAMPLE

(i) $ae^x + bx \tan x + c = 0$

(ii) $3x^2 - \log x^2 + \sin x + e^x - 1 = 0$

Solution of an equation $f(x) = 0$ means we have to find its roots or zeros.

Before we develop various numerical **methods** we shall list below some of the basic properties of an algebraic equation.

(i) Every algebraic equation of nth degree, where n is a positive integer, has n and only n roots.

(ii) Complex roots occur in pairs, i.e. if $(a + ib)$ is a root of $f(x) = 0$, then $a - ib$ is also a root of this equation.

(iii) If $x = \alpha$ is a root of $f(x) = 0$, a polynomial of degree n, then $(x - \alpha)$ is a factor of $f(x)$. On dividing $f(x)$ by $(x - \alpha)$, we obtain a polynomial of degree $(n - 1)$.

(iv) *Descarte's rule of signs:* The number of positive roots of an equation $f(x) = 0$ with real coefficients cannot exceed the number of changes in sign of the coefficients in the polynomial $f(x) = 0$. Similarly, the number of negative roots of $f(x) = 0$ cannot exceed the number of changes in the sign of the coefficients of $f(-x) = 0$, for example consider the equation

$$f(x) = 2x^3 - 8x^2 + 5x - 6 = 0$$

Signs of $f(x)$ are: $+ \quad - \quad + \quad -$

There are three changes in sign, the given equation may have three positive roots.

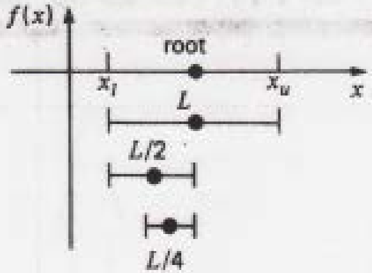
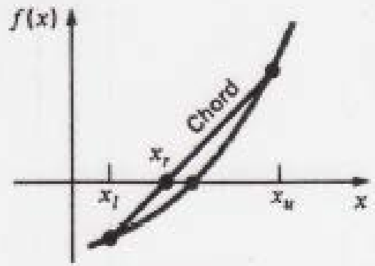
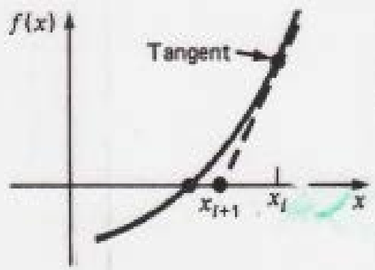
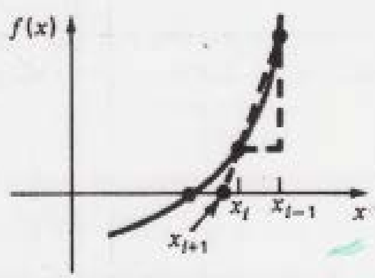
Two Fundamental Approaches

Bracketing Methods

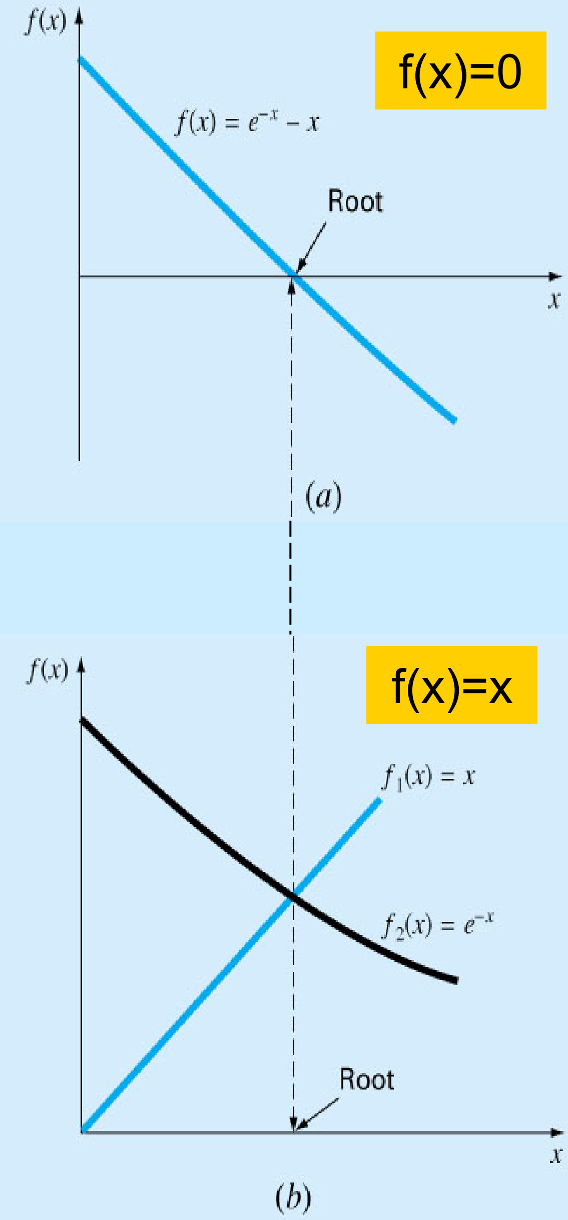
- Bisection
- False Position Approach

Open Methods

- Fixed-Point Iteration
- Newton-Raphson
- Secant Method

Method	Formulation	Graphical Interpretation	Errors and Stopping Criteria
		Bracketing methods:	
Bisection	$x_r = \dfrac{x_l + x_u}{2}$ If $f(x_l)f(x_r) < 0,\ x_u = x_r$ $f(x_l)f(x_r) > 0,\ x_l = x_r$		Stopping criterion: $\left\| \dfrac{x_r^{new} - x_r^{old}}{x_r^{new}} \right\| 100\% \le \epsilon_s$
False Position	$x_r = x_u - \dfrac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)}$ If $f(x_l)f(x_r) < 0,\ x_u = x_r$ $f(x_l)f(x_r) > 0,\ x_l = x_r$		Stopping criterion: $\left\| \dfrac{x_r^{new} - x_r^{old}}{x_r^{new}} \right\| 100\% \le \epsilon_s$
		Open methods:	
Newton-Raphson	$x_{i+1} = x_i - \dfrac{f(x_i)}{f'(x_i)}$		Stopping criterion: $\left\| \dfrac{x_{i+1} - x_i}{x_{i+1}} \right\| 100\% \le \epsilon_s$ Error: $E_{i+1} = 0(E_i^2)$
Secant	$x_{i+1} = x_i - \dfrac{f(x_i)(x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)}$		Stopping criterion: $\left\| \dfrac{x_{i+1} - x_i}{x_{i+1}} \right\| 100\% \le \epsilon_s$

Iteration



Bracketing Methods

- **Both bisection and false-position methods require the root to be bracketed by the endpoints.**

- **How to find the endpoints?**

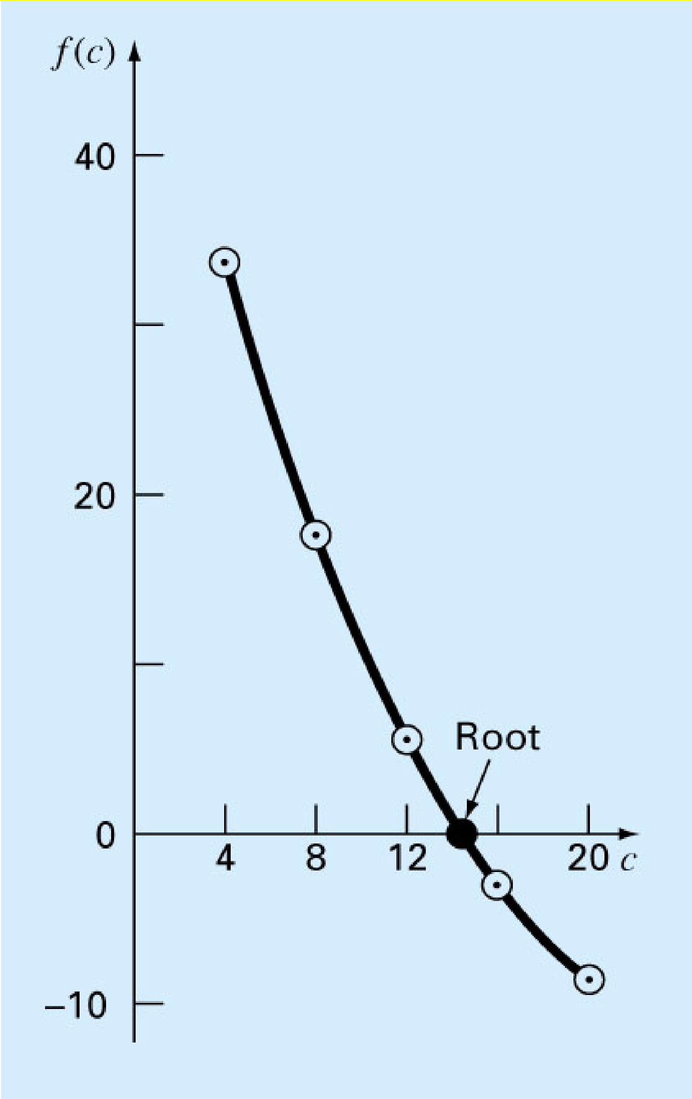
  *** plotting the function**

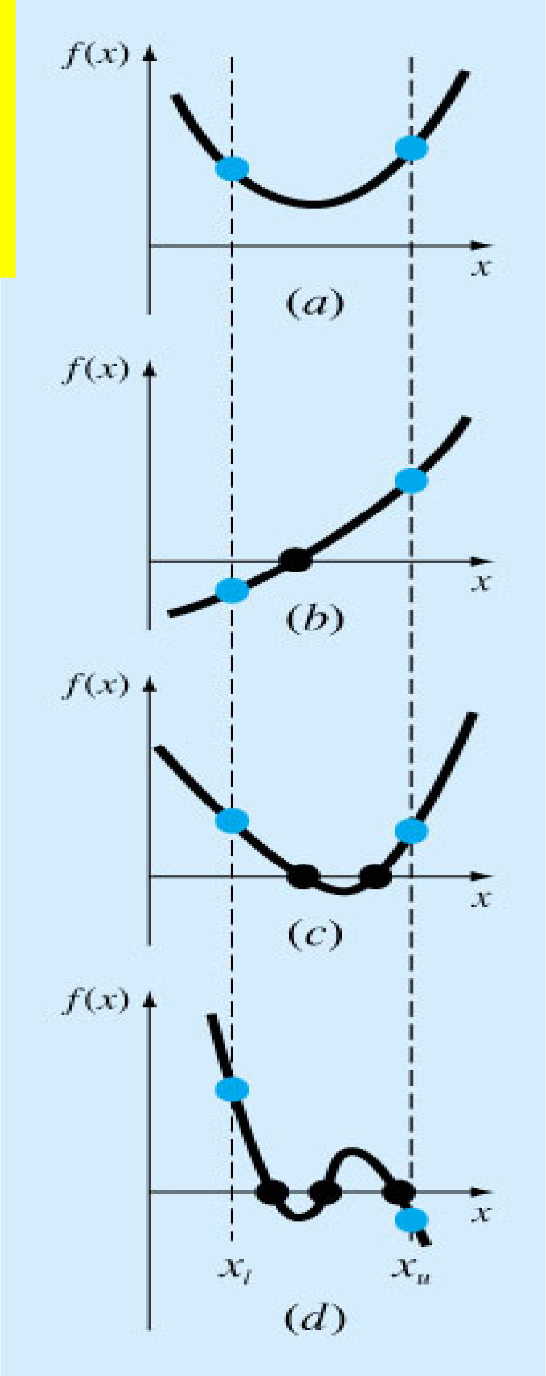
  *** incremental search**

  *** trial and error**

- **Graphic Methods** (Rough Estimation)
- **Single Root** e.g.(X-1) (X-2) = 0 (X = 1, X =2)
- **Double Root** e.g. (X-1)^2 = 0 (X = 1)
- Effective Only to **Single Root Cases**

Graphical methods





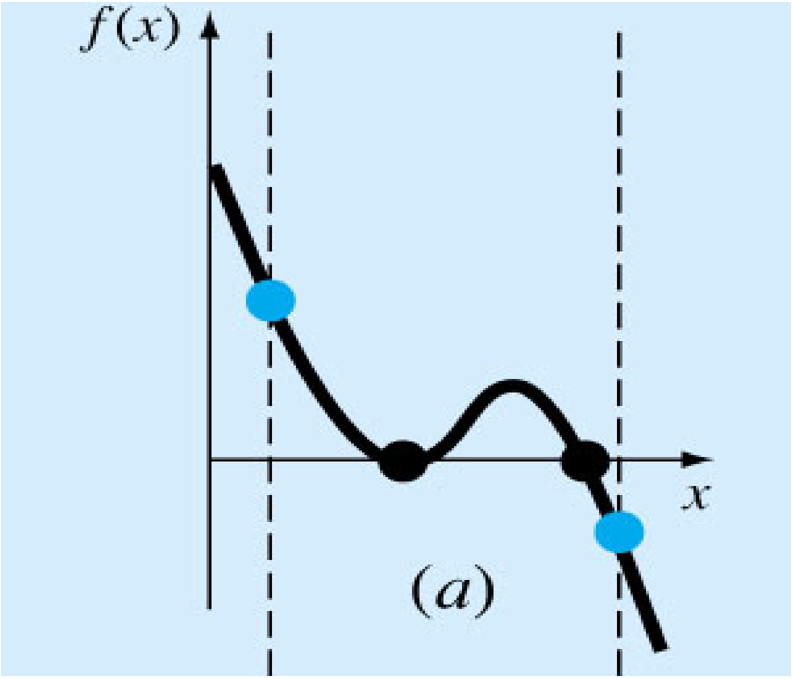
No real root
(same sign)

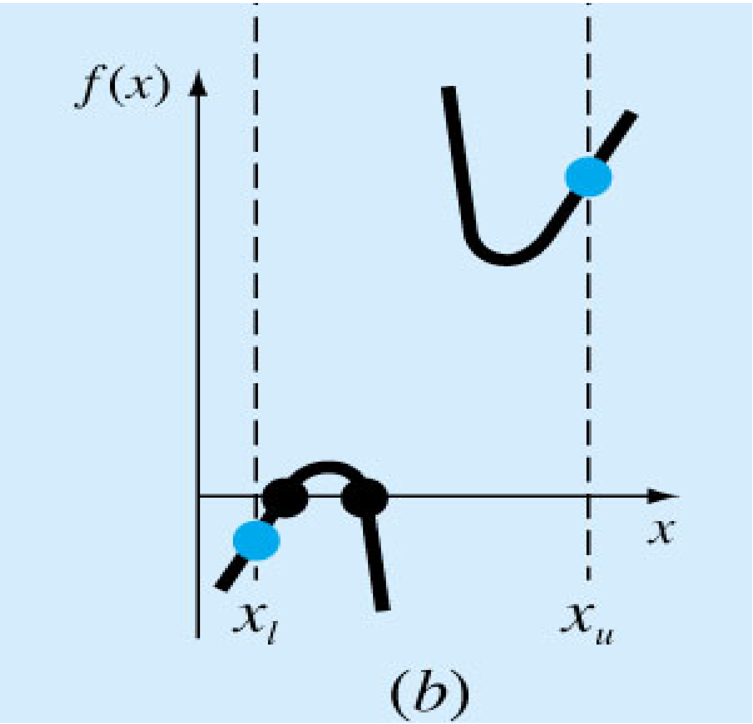
Single root
(change sign)

Two roots
(same sign)

Three roots
(change sign)

Special Cases



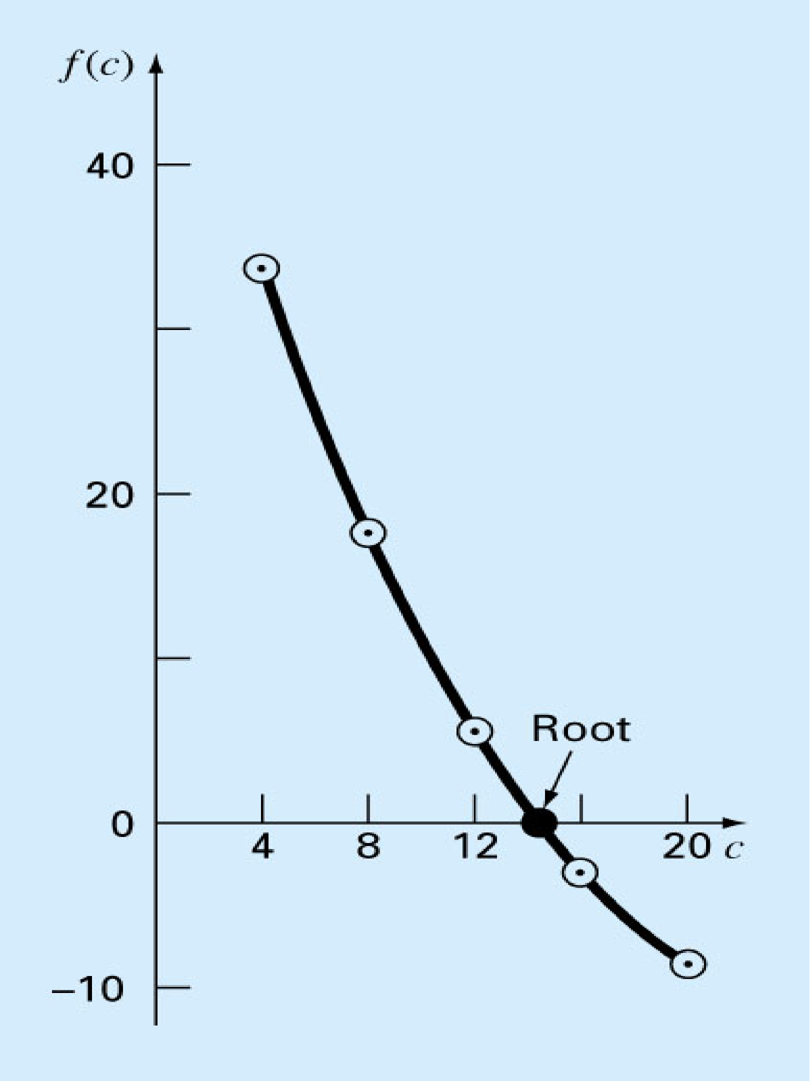


Multiple Roots **Discontinuity**

Bracketing Methods

- Two initial guesses for the root are required. These guesses must “bracket” or be on either side of the root.

- If one root of a real and continuous function, $f(x)=0$, is bounded by values $x=x_l$, $x=x_u$ then $f(x_l)*f(x_u) < 0$.
(The function changes sign on opposite sides of the root)



Bisection Method

Theorem An equation f(x)=0, where f(x) is a real continuous function, has at least one root between x_l and x_u if $f(x_l)\, f(x_u) < 0$.

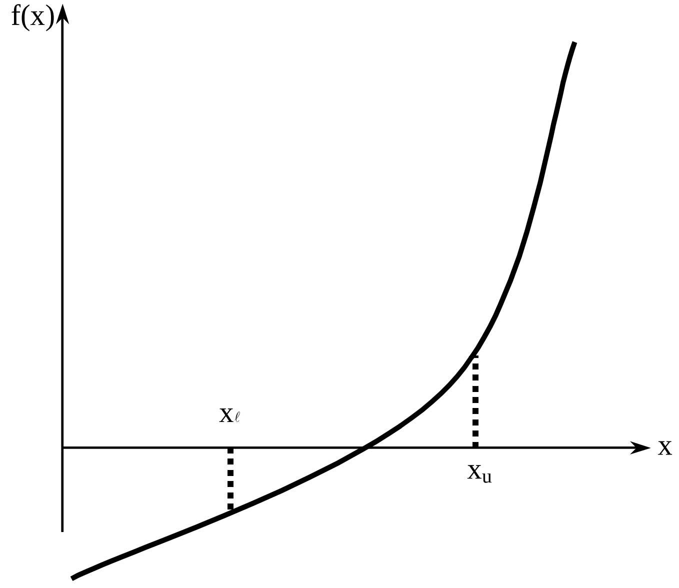


Figure 1 At least one root exists between the two points if the function is real, continuous, and changes sign.

Basis of Bisection Method

Theorem If function $f(x)$ in $f(x) = 0$ does not change sign between two points, roots may still exist between the two points.

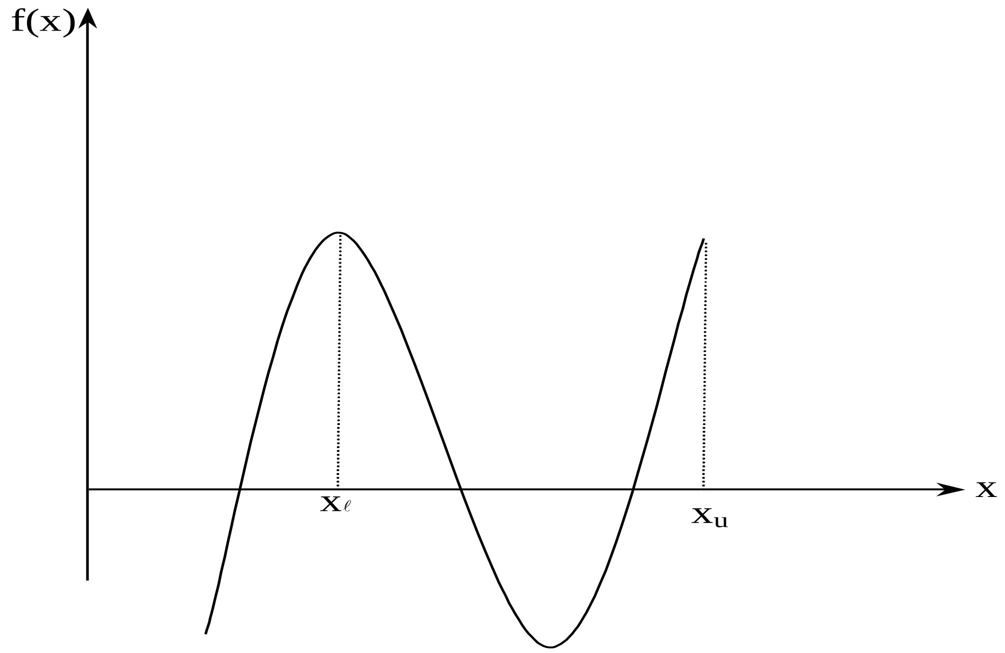


Figure 2 If function does not change sign between two points, roots of the equation may still exist between the two points.

Basis of Bisection Method

Theorem If the function $f(x)$ in $f(x) = 0$ does not change sign between two points, there may not be any roots between the two points.

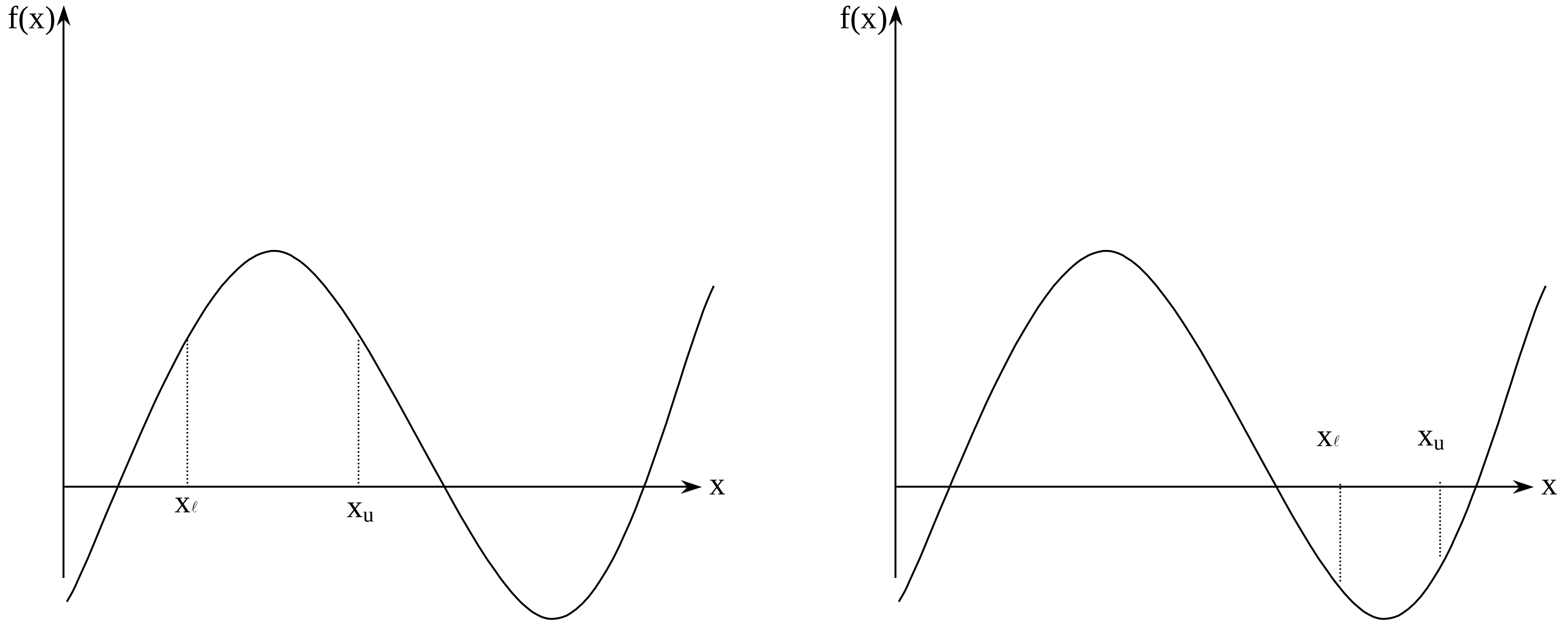


Figure 3 If the function does not change sign between two points, there may not be any roots for the equation between the two points.

Basis of Bisection Method

Theorem If the function $f(x)$ in $f(x) = 0$ changes sign between two points, more than one root may exist between the two points.

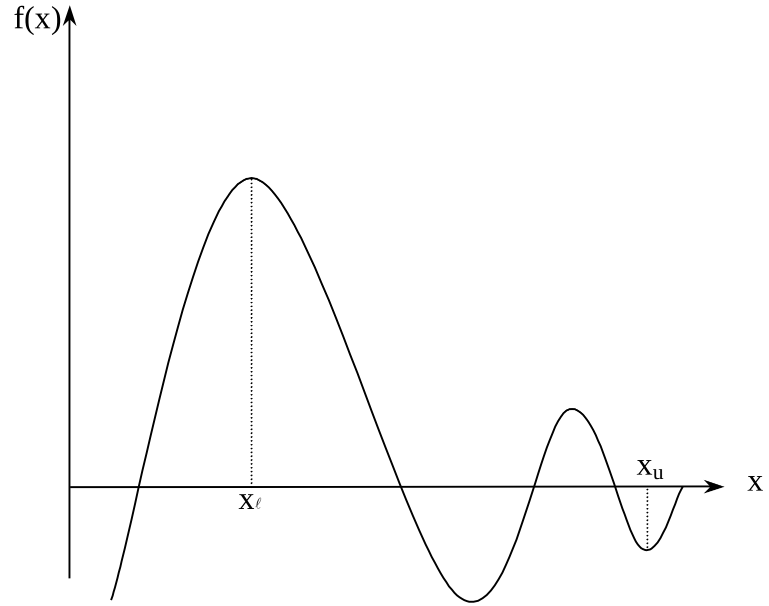


Figure 4 If the function changes sign between two points, more than one root for the equation may exist between the two points.

Bisection Method

- Step 1: Choose x_l and x_u such that x_l and x_u bracket the root, i.e. $\mathbf{f(x_l)*f(x_u) < 0}$.
- Step 2 : Estimate the root (bisection).

$$\boxed{x_r = 0.5*(x_l + x_u)}$$

- Step 3: Determine the new bracket.

  If $\mathbf{f(x_r)*f(x_l) < 0}$ $\mathbf{x_u = x_r}$

  else $\mathbf{x_l = x_r}$ end

- Step 4: Examine if $\mathbf{x_r}$ is the root.
- Step 5: If not, Repeat steps 2 and 3 until convergence

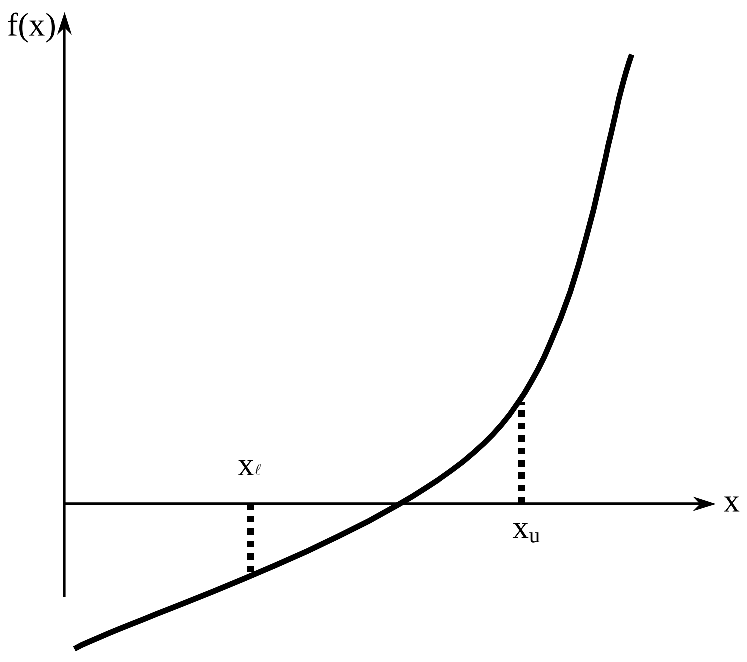
$$(a)\quad f(x_r) \approx 0,\ i.e.,\ \left|f(x_r)\right| \le \varepsilon$$

$$(b)\quad \varepsilon_a = \left|\frac{x_r^{new} - x_r^{old}}{x_r^{new}}\right|(100\%) \le \varepsilon_s \ \text{in successive iterations}$$

$$(c)\quad \text{the maximum number of iterations has been reached}$$

Step 1

Choose x_l and x_u as two guesses for the root such that $f(x_l)\,f(x_u) < 0$, or in other words, $f(x)$ changes sign between x_l and x_u. This was demonstrated in the Figure.



Figure

Step 2

Estimate the root, x_r of the equation f(x) = 0 as the mid point between x_l and x_u as

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

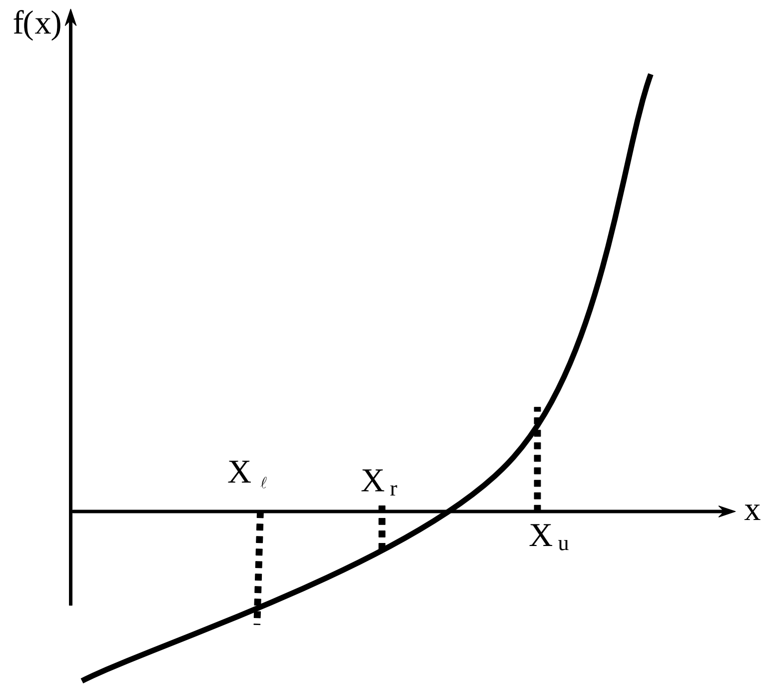


Figure Estimate of x_r

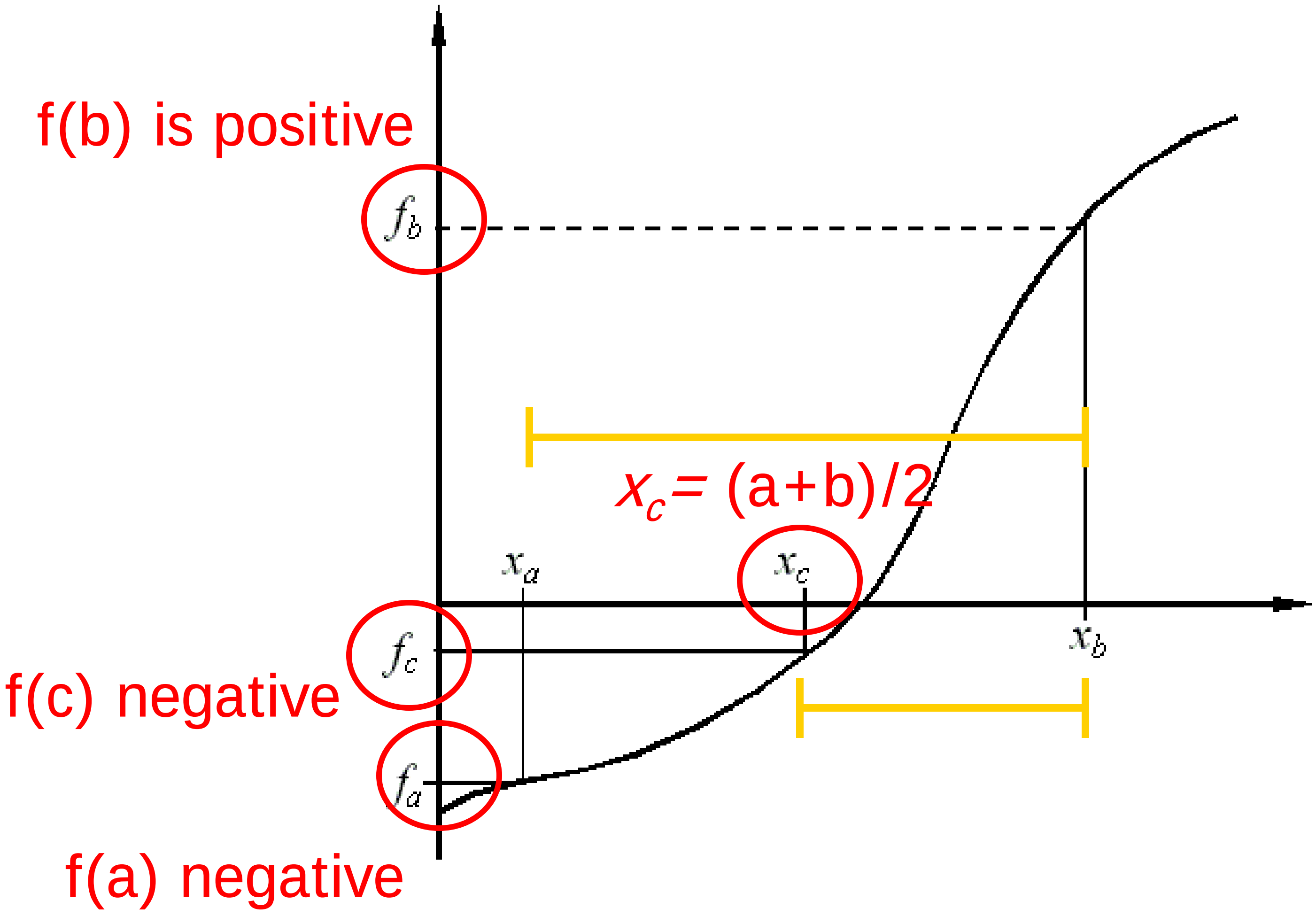
Step 3

Now check the following

a) If $f(x_l)f(x_r)<0$, then the root lies between x_l and x_r; then $x_l = x_l$; $x_u = x_r$.

b) If $f(x_l)f(x_r)>0$, then the root lies between x_r and x_u; then $x_l = x_r$; $x_u = x_u$.

c) If $f(x_l)f(x_r)=0$, then the root is $x_{r.}$ Stop the algorithm if this is true.



What will be the next interval?

Step 4

Find the new estimate of the root

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

Find the relative approximate error

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

where

$$x_r^{old} = \text{previous estimate of root}$$

$$x_r^{new} = \text{current estimate of root}$$

Step 5

Compare the absolute relative approximate error $|\in_a|$ with the pre-specified error tolerance $\in_s$.

Is $|\in_a| > \in_s$?

Yes → Go to Step 2 using new upper and lower guesses.

No → Stop the algorithm

Note one should also check whether the number of iterations is more than the maximum number of iterations allowed. If so, one needs to terminate the algorithm and notify the user about it.

Example 1

To aid in the understanding of how this method works to find the root of an equation, the graph of f(x) is shown to the right, where

$$f(x) = x^3 - 0.165x^2 + 3.993\times10^{-4}$$

Let us assume $x_\ell = 0.00$

$x_u = 0.11$

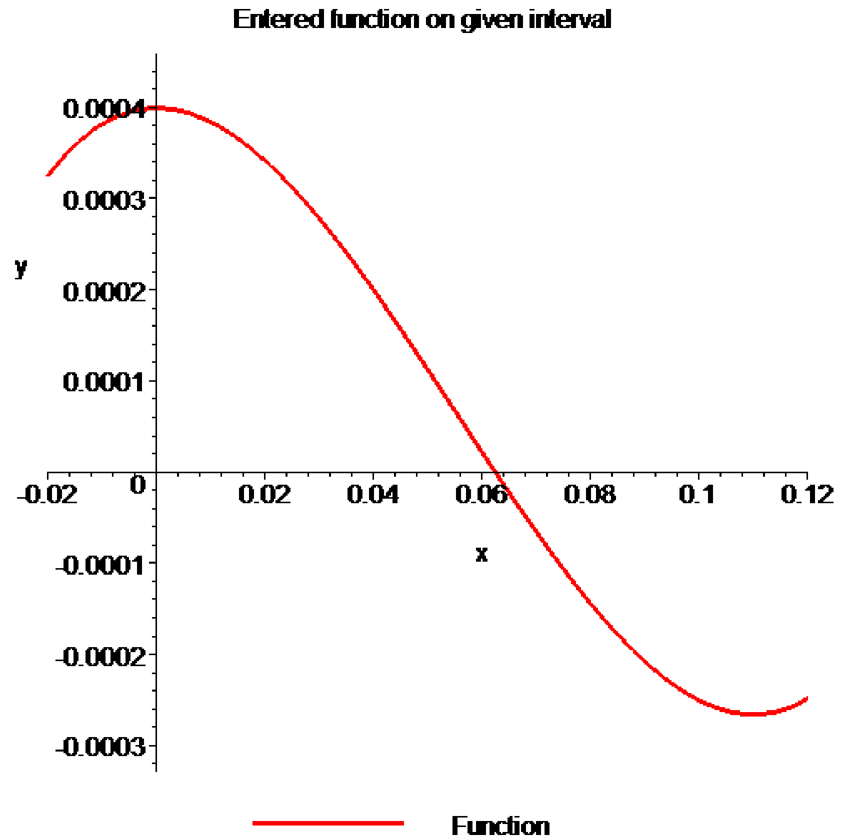


Figure Graph of the function f(x)

Check if the function changes sign between x_l and x_u .

$$f(x_l) = f(0) = (0)^3 - 0.165(0)^2 + 3.993\times10^{-4} = 3.993\times10^{-4}$$

$$f(x_u) = f(0.11) = (0.11)^3 - 0.165(0.11)^2 + 3.993\times10^{-4} = -2.662\times10^{-4}$$

Hence $$f(x_l)f(x_u) = f(0)f(0.11) = \left(3.993\times10^{-4}\right)\left(-2.662\times10^{-4}\right) < 0$$

So there is at least on root between x_l and x_u, that is between 0 and 0.11

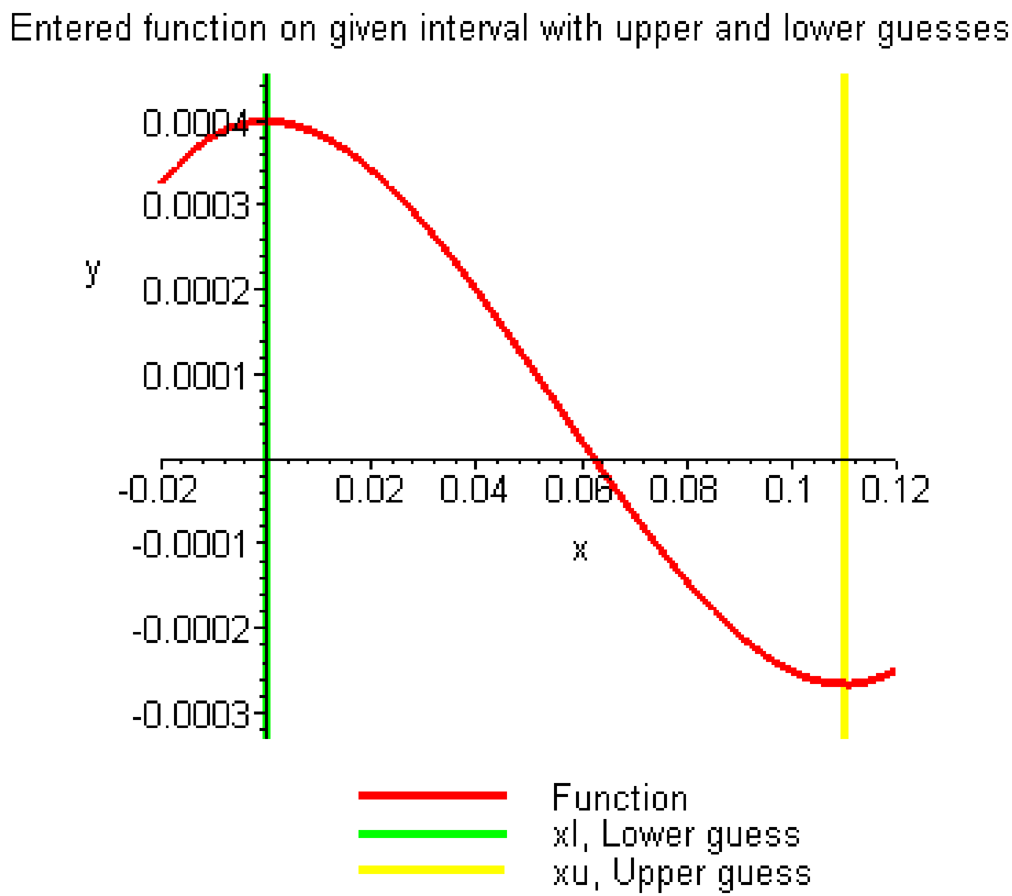


Figure Graph demonstrating sign change between initial limits

Iteration 1

The estimate of the root is $$x_r = \frac{x_\ell + x_u}{2} = \frac{0+0.11}{2} = 0.055$$

$$f(x_r) = f(0.055) = (0.055)^3 - 0.165(0.055)^2 + 3.993\times10^{-4} = 6.655\times10^{-5}$$

$$f(x_l)f(x_r) = f(0)f(0.055) = \left(3.993\times10^{-4}\right)\left(6.655\times10^{-5}\right) > 0$$

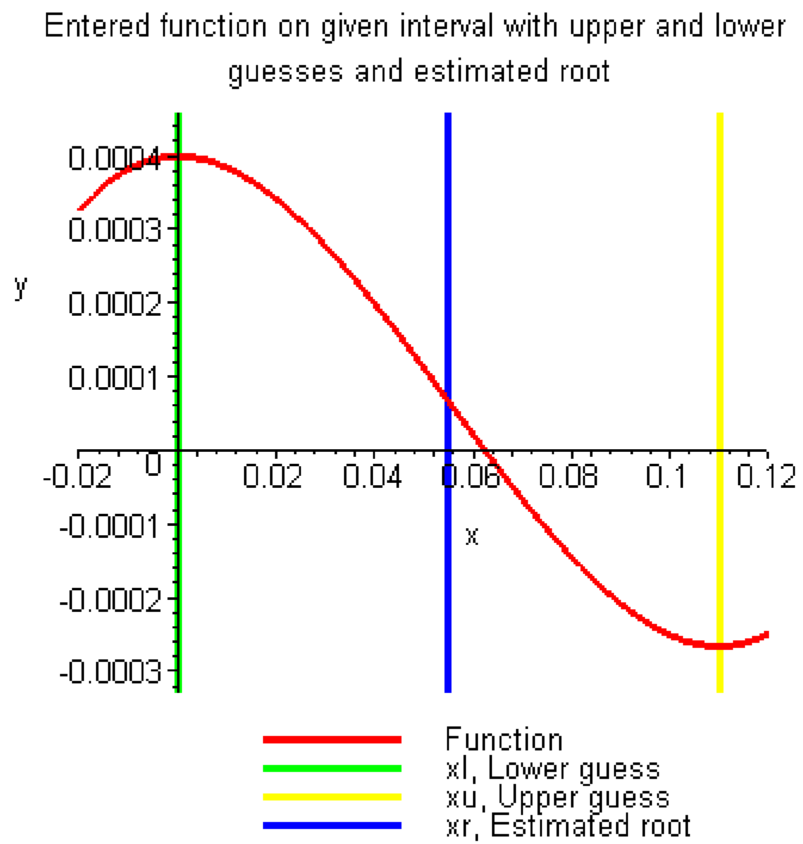


Figure Estimate of the root for Iteration 1

Hence the root is bracketed between x_r and x_u, that is, between 0.055 and 0.11. So, the lower and upper limits of the new bracket are $x_l = 0.055, \; x_u = 0.11$

At this point, the absolute relative approximate error $|\in_a|$ cannot be calculated as we do not have a previous approximation.

Iteration 2

The estimate of the root is $x_r = \dfrac{x_\ell + x_u}{2} = \dfrac{0.055 + 0.11}{2} = 0.0825$

$$f(x_r) = f(0.0825) = (0.0825)^3 - 0.165(0.0825)^2 + 3.993 \times 10^{-4} = -1.622 \times 10^{-4}$$

$$f(x_l) f(x_r) = f(0.055) f(0.0825) = (-1.622 \times 10^{-4})(6.655 \times 10^{-5}) < 0$$

Hence the root is bracketed between x_l and x_r, that is, between 0.055 and 0.0825. So, the lower and upper limits of the new bracket are $x_l = 0.055, \; x_u = 0.0825$

The absolute relative approximate error $|\in_a|$ at the end of Iteration 2 is

$$|\in_a| = \left| \frac{x_r^{new} - x_r^{old}}{x_r^{new}} \right| \times 100 = \left| \frac{0.0825 - 0.055}{0.0825} \right| \times 100 = 33.333\%$$

None of the significant digits are at least correct in the estimate root of $x_m = 0.0825$ because the absolute relative approximate error is greater than 5%.

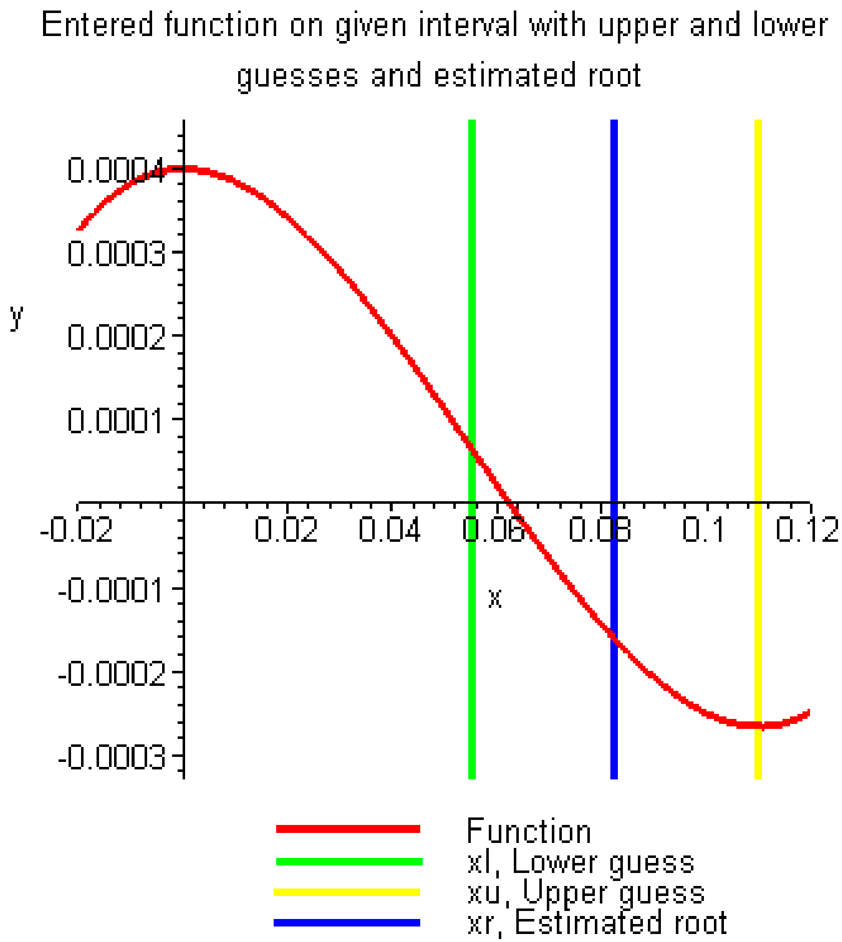


Figure Estimate of the root for Iteration 2

Iteration 3

The estimate of the root is $x_r = \dfrac{x_\ell + x_u}{2} = \dfrac{0.055 + 0.0825}{2} = 0.06875$

$$f(x_r) = f(0.06875) = (0.06875)^3 - 0.165(0.06875)^2 + 3.993\times10^{-4} = -5.563\times10^{-5}$$

$$f(x_l)f(x_r) = f(0.055)f(0.06875) = (6.655\times10^{-5})(-5.563\times10^{-5}) < 0$$

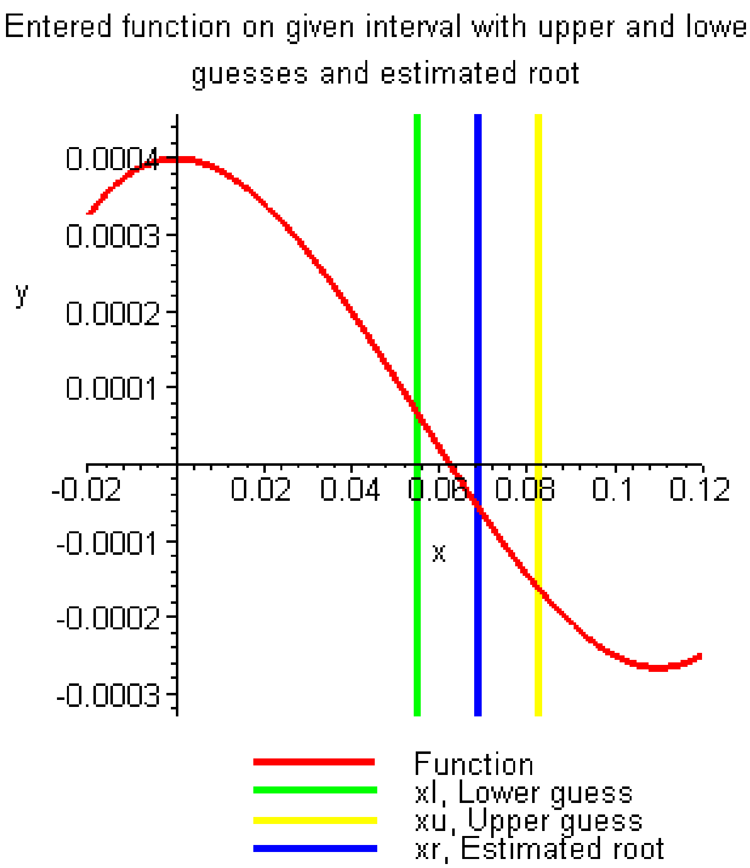


Figure Estimate of the root for Iteration 3

Hence the root is bracketed between x_ℓ and x_r , that is, between 0.055 and 0.06875. So, the lower and upper limits of the new bracket are

$$x_l = 0.055, \;\; x_u = 0.06875$$

The relative approximate error $|\in_a|$ at the end of Iteration 3 is

$$|\in_a| = \left|\frac{x_r^{new} - x_r^{old}}{x_r^{new}}\right| \times 100 = \left|\frac{0.06875 - 0.0825}{0.06875}\right| \times 100 = 20\%$$

Still none of the significant digits are at least correct in the estimated root of the equation as the absolute relative approximate error is greater than 5%. Seven more iterations were conducted and these iterations are shown in the **Table**.

Hence the number of significant digits at least correct is given by the largest value or *m* for which

$$|\in_a| \le 0.5 \times 10^{2-m} \Rightarrow 0.1721 \le 0.5 \times 10^{2-m}$$

$$0.3442 \le 10^{2-m} \Rightarrow \log(0.3442) \le 2 - m$$

$$m \le 2 - \log(0.3442) = 2.463$$

Table Root of f(x)=0 as function of number of iterations for bisection method.

Iteration	x_ℓ	x_u	x_r	$\left\|\in_a\right\|$ %	$f(x_r)$
1	0.00000	0.11	0.055	----------	6.655×10^{-5}
2	0.055	0.11	0.0825	33.33	-1.622×10^{-4}
3	0.055	0.0825	0.06875	20.00	-5.563×10^{-5}
4	0.055	0.06875	0.06188	11.11	4.484×10^{-6}
5	0.06188	0.06875	0.06531	5.263	-2.593×10^{-5}
6	0.06188	0.06531	0.06359	2.702	-1.0804×10^{-5}
7	0.06188	0.06359	0.06273	1.370	-3.176×10^{-6}
8	0.06188	0.06273	0.0623	0.6897	6.497×10^{-7}
9	0.0623	0.06273	0.06252	0.3436	-1.265×10^{-6}
10	0.0623	0.06252	0.06241	0.1721	-3.0768×10^{-7}

So $m = 2$

The number of significant digits at least correct in the estimated root of 0.06241 at the end of the 10^{th} iteration is 2.

Example 2

Find the root of the equation $x^3 + 4x^2 - 1 = 0$.

Solution Let, $a = 0$ and $b = 1$.

Now, $f(0) = (0)^3 + 4(0)^2 - 1 = -1 < 0$ and $f(1) = (1)^3 + 4(1)^2 - 1 = 4 > 0$.
$f(a)$ and $f(b)$ has opposite sign. Therefore, $f(x)$ has a root in the interval $[a, b] = [0, 1]$ $x_c = (0 + 1)/2 = 0.5$, $f(0.5) = 0.125$.

Now, $f(a)$ and $f(x_c)$ has opposite sign. So, the next interval is [0, 0.5]

a	***b***	**x_c = (a+b)/2**	***f*(*a*)**	***f*(*b*)**	***f*(x_c)**
0	1	0.5	**-1**	4	**0.125**
0	0.5	0.25	-1	**0.125**	**- 0.73438**
0.25	0.5	0.375	-0.73438	**0.125**	**- 0.38477**
0.375	0.5	0.4375	-0.38477	**0.125**	**- 0.15063**
0.4375	0.5	0.46875	-0.15063	**0.125**	**- 0.0181**
0.46875	0.5	0.484375	**-0.0181**	0.125	**0.05212**
0.46875	0.484375	0.476563	**- 0.0181**	0.05212	**0.01668**

... and so we approach the root 0.472834.

Hand Calculation Example

Bisection Method

$$Example:\ f(x) = x^2 - 2x - 3 = 0$$

$$initial\ estimeates\ [x_l, x_u] = [2.0,\ 3.2]$$

$iter$	x_l	x_u	x_r	$f(x_r)$	Δx
1	2.0	3.2	2.6	−1.44	1.2
2	2.6	3.2	2.9	−0.39	0.6
3	2.9	3.2	3.05	0.2025	0.3
4	2.9	3.05	2.975	−0.0994	0.15
5	2.975	3.05	3.0125	0.0502	0.075
6	2.975	3.0125	2.99375	−0.02496	0.0375

$f(2) = -3$, $f(3.2) = 0.84$

Bisection Method

Advantages:

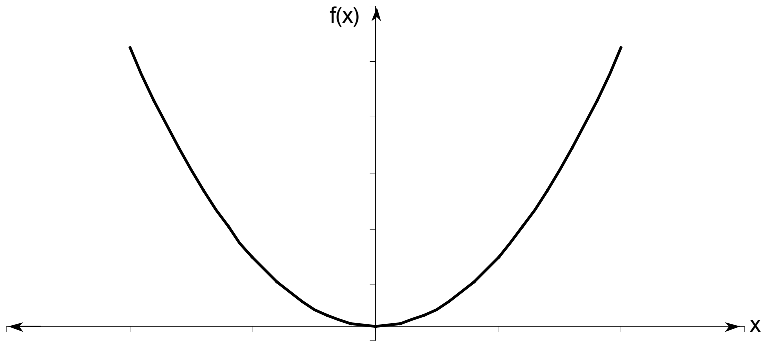
- Simple and easy to implement
- One function evaluation per iteration
- No knowledge of the derivative is needed
- Always convergent, the root bracket gets halved with each iteration.

Disadvantages:

- We need two initial guesses *a* and *b* which bracket the root.
- Slowest method to converge to the solution.
- If one of the initial guesses is close to the root, the convergence is slower
- When an interval contains more than one root, the bisection method can find *only* one of them.

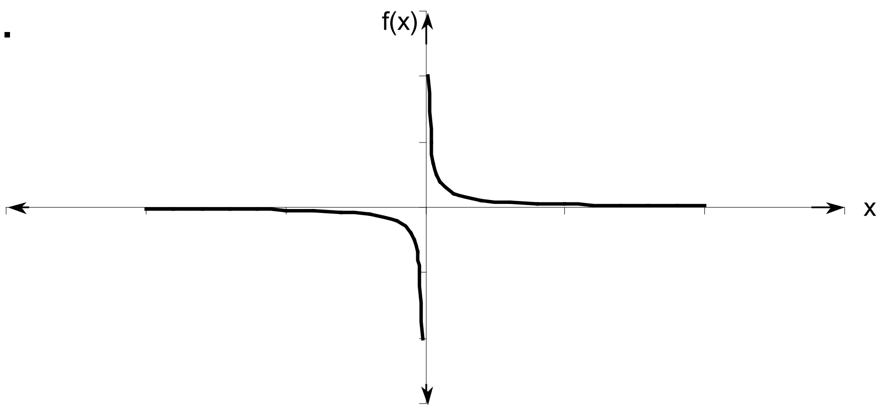
Drawbacks

- If a function f(x) is such that it just touches the x-axis it will be unable to find the lower and upper guesses.



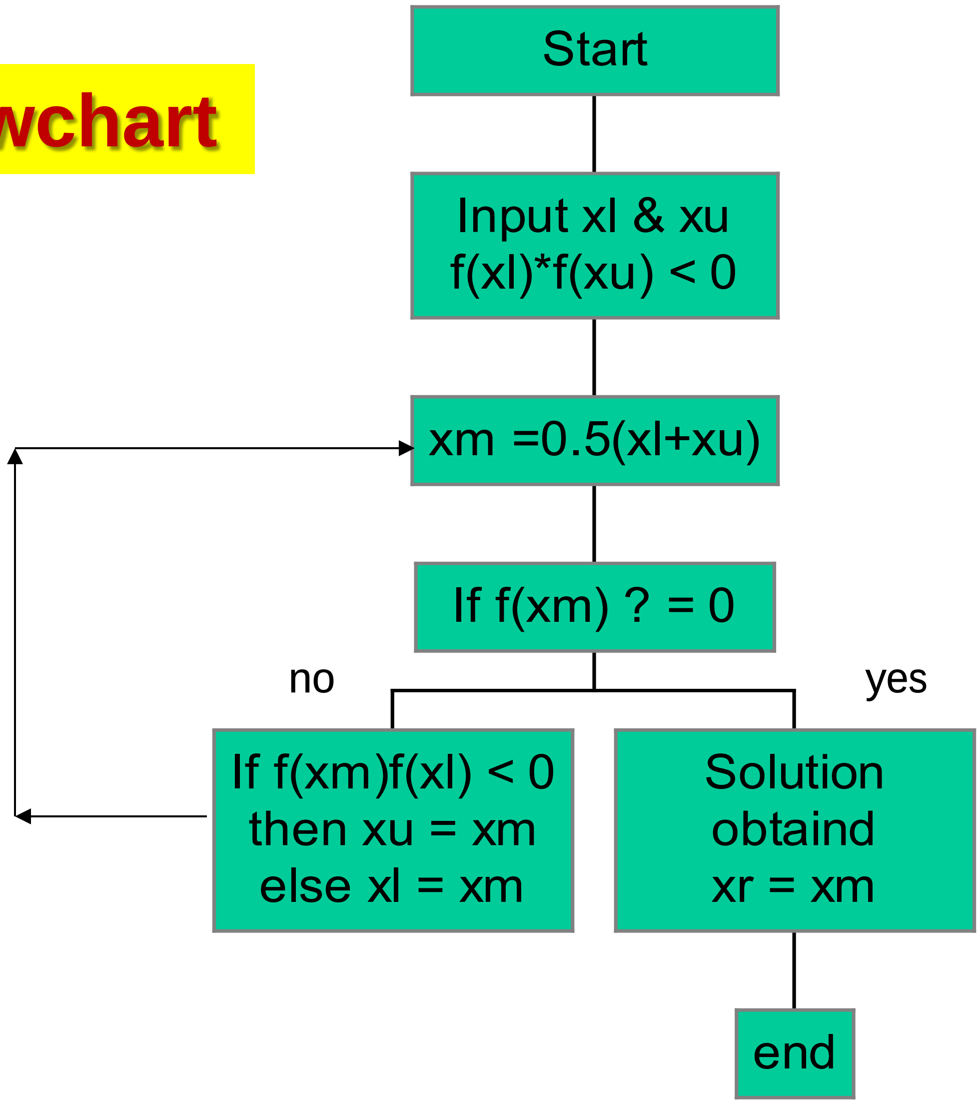
$$f(x) = x^2$$

- If the function $f(x)$ is not continuous between a and b, but $f(a)$ and $f(b)$ has opposite signs, then there may not exist any root between a and b.



$$f(x) = \frac{1}{x}$$

Bisection Flowchart



```
function root = bisection(func,xl,xu,es,maxit)
% bisection(func,xl,xu,es,maxit);
%   use bisection method to find the root of a function
% input:
%   func = name of function
%   xl, xu = lower and upper guesses
%   es = (optional) stopping criterion (%)
%   maxit = (optional) maximum allowable iterations
% output:
%   root = real root

if func(xl)*func(xu) > 0 % if guesses do not bracket a sign change,
    error('no bracket')  % display an error message and terminate
    return
end
% if necessary, assign default values of maxit and es
if nargin < 5, maxit = 50; end  % if maxit blank, set to 50
if nargin < 4, es = 0.001; end  % if es blank, set to 0.001

%bisection
iter = 0;
xr = xl;
while (1)
    xrold = xr;
    xr = (xl + xu)/2;
    iter = iter + 1;
    if xr ~= 0, ea =abs((xr-xrold)/xr) * 100; end
    test = func(xl) * func(xr);
    if test < 0
        xu = xr;
    elseif test > 0
        xl = xr;
    else
        ea = 0;
    end
    if ea <= es | iter >= maxit, break, end
end
root = xr;
```

M-file in textbook

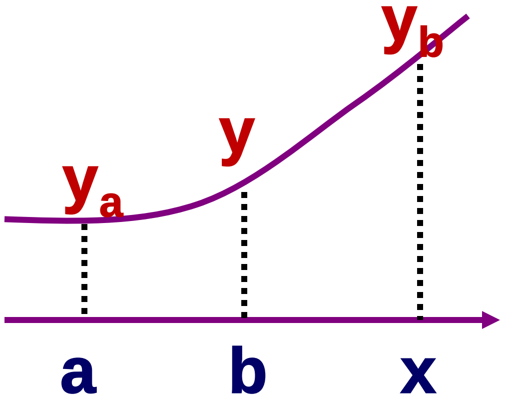
```
function [x,y] = bisect2(func)

% Find root near x1 using the bisection method.
% Input:  func     string containing name of function
%         xl,xu    initial guesses
%         es       allowable tolerance in computed root
%         maxit    maximum number of iterations
% Output: x        row vector of approximations to root

xl = input('enter lower bound xl = ');
xu = input('enter upper bound xu = ');
es = input('allowable tolerance es = ');
maxit = input('maximum number of iterations maxit = ');
a(1) = xl; b(1) = xu;
ya(1) = feval(func,a(1)); yb(1) = feval(func,b(1));
if ya(1) * yb(1) > 0.0
     error('Function has same sign at end points')
end
for i = 1 : maxit
     x(i) = (a(i) + b(i))/2;  y(i) = feval(func, x(i));
     if((x(i) - a(i)) < es)
        disp('Bisection method has converged'); break;
     end
     if y(i) == 0.0
        disp('exact zero found'); break;
     elseif y(i) * ya(i) < 0
        a(i+1) = a(i); ya(i+1) = ya(i);
        b(i+1) = x(i); yb(i+1) = y(i);
     else
        a(i+1) = x(i); ya(i+1) = y(i);
        b(i+1) = b(i); yb(i+1) = yb(i);
     end;
     iter = i;
end
if(iter >= maxit)
        disp('zero not found to desired tolerance');
end
n = length(x); k = 1:n;
out = [k'  a(1:n)'  b(1:n)'    x'      y'];
disp('    step       xl         xu         xr      f(xr)')
disp(out)
```

An interactive M-file

Use “feval” to evaluate the function“func”



break: terminate a “for” or “while” loop

Exercises

1. Find the real root of the equation $f(x)=x^3 - x - 1 = 0$ correct to 2 decimal places.($\varepsilon = 0.01$).

   Answer: 1.325683

2. Find the real root of the equation $f(x)=x^4 - \cos(x) + x = 0$ correct to 2 decimal places.($\varepsilon = 0.01$).

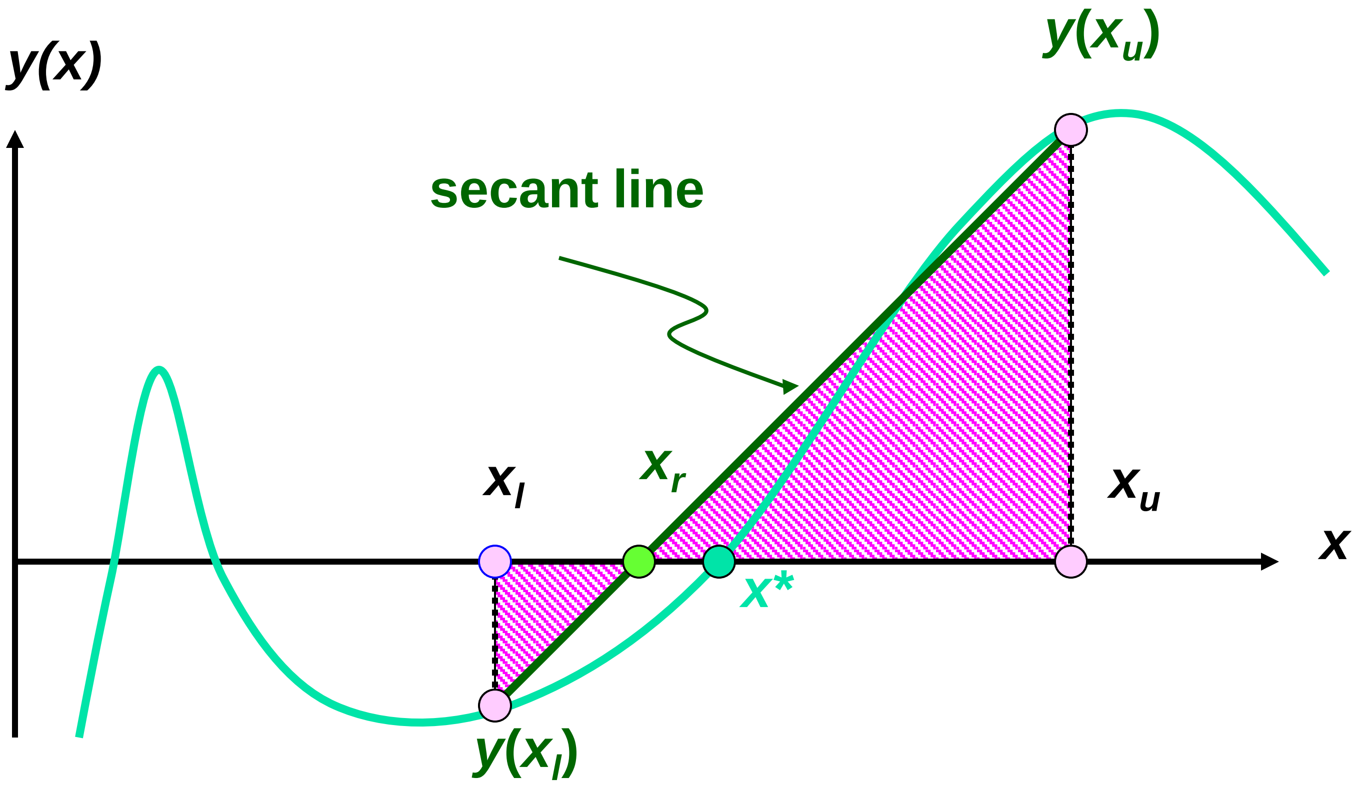
   Answer: 0.637695

False-Position (point) Method

Why bother with another method?

- **The bisection method is simple and guaranteed to converge (single root) but the convergence is slow and non-monotonic!**
- **The bisection method is a brute force method and makes no use of information about the function**
- **Bisection only the sign, not the value $f(x_k)$ itself**
- **False-position method takes advantage of function curve shape <u>may</u> converge more quickly**

False-Position Method



Straight line (linear) approximation to exact curve

Based on two similar triangles, shown in the Figure, one gets:

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

$$f(x_L) < 0;\ x_r - x_L > 0$$
$$f(x_U) > 0;\ x_r - x_U < 0$$

The signs for both sides of the Eq. is consistent :

From above, one obtains

$$(x_r - x_L) f(x_U) = (x_r - x_U) f(x_L)$$

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

The above equation can be solved to obtain the next predicted root x_r , as

$$\boxed{x_r = \frac{x_U f(x_L) - x_L f(x_U)}{f(x_L) - f(x_U)} \quad \text{or} \quad x_r = x_U - \frac{f(x_U)\{x_L - x_U\}}{f(x_L) - f(x_U)}}$$

Step-By-Step False-Position Algorithms

1. Choose x_L and x_U as two guesses for the root such that $f(x_L)f(x_U) < 0$

2. Estimate the root, $x_r = \dfrac{x_U f(x_L) - x_L f(x_U)}{f(x_L) - f(x_U)}$

3. Now check the following

(a) If $f(x_L)f(x_r) < 0$, then the root lies between x_L and x_r ; then $x_L = x_L$ and $x_U = x_r$

(b) If $f(x_L)f(x_r) > 0$, then the root lies between x_r and x_U ; then $x_L = x_r$ and $x_U = x_U$

(c) If $f(x_L)f(x_r) = 0$, then the root is x_r.

Stop the algorithm if this is true.

4. Find the new estimate of the root

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

Find the relative approximate error as

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

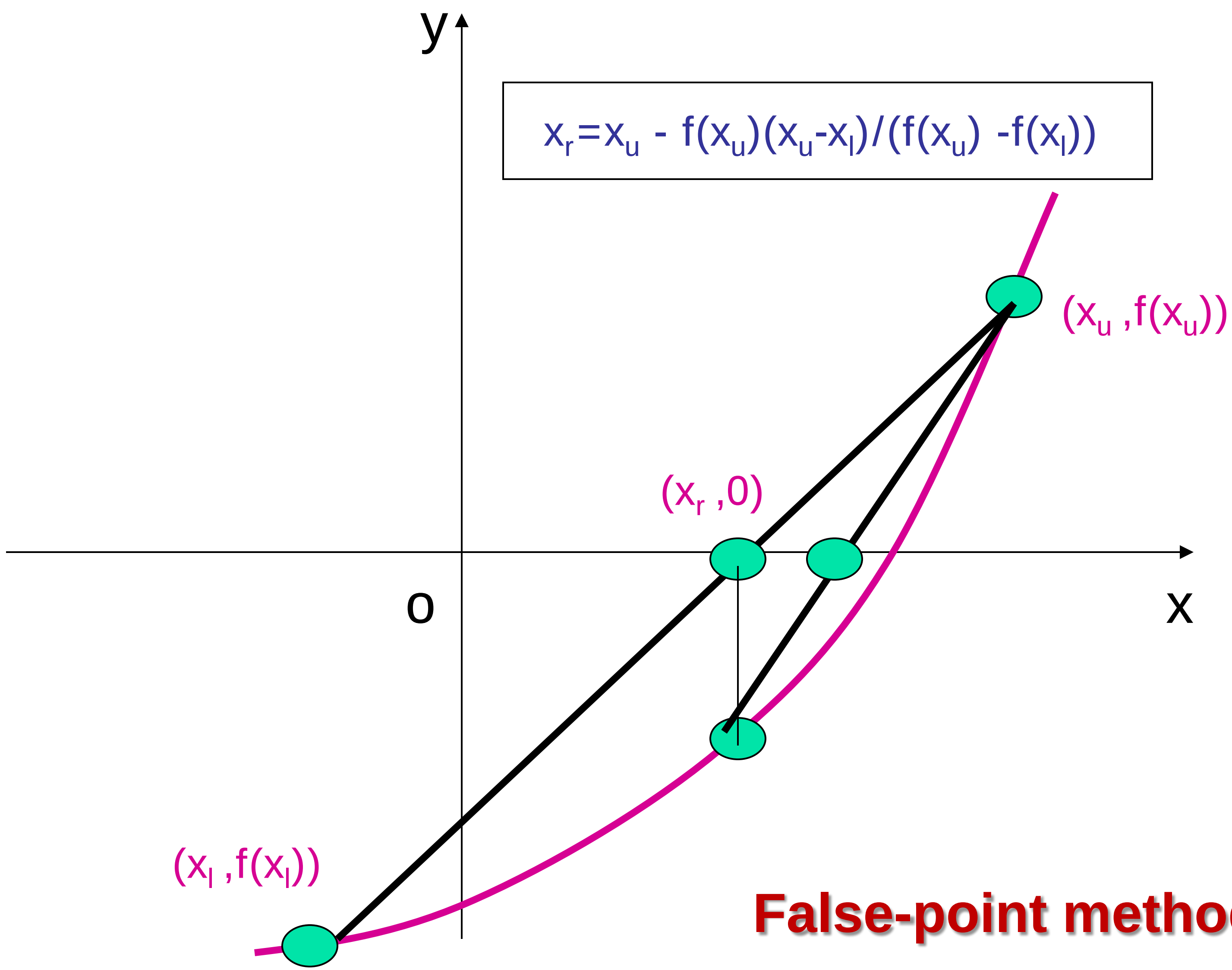
where

x_r^{new} = estimated root from present iteration

x_r^{old} = estimated root from previous iteration

5. $say \in_s = 10^{-3} = 0.001.$ If $|\in_a| > \in_s$, then go to step 3, else stop the algorithm.

Notes: The False-Position and Bisection algorithms are quite similar. The only difference is the formula used to calculate the new estimate of the root x_r



False-point method

Example 1

The floating ball has a specific gravity of 0.6 and has a radius of 5.5cm. You are asked to find the depth to which the ball is submerged when floating in water. The equation that gives the depth x to which the ball is submerged under water is given by

$$x^3 - 0.165x^2 + 3.993\times10^{-4} = 0$$

Use the false-position method of finding roots of equations to find the depth x to which the ball is submerged under water. Conduct three iterations to estimate the root of the above equation. Find the absolute relative approximate error at the end of each iteration, and the number of significant digits at least correct at the converged iteration.

Solution From the physics of the problem

$$0 \le x \le 2R$$

$$0 \le x \le 2(0.055)$$

$$0 \le x \le 0.11$$

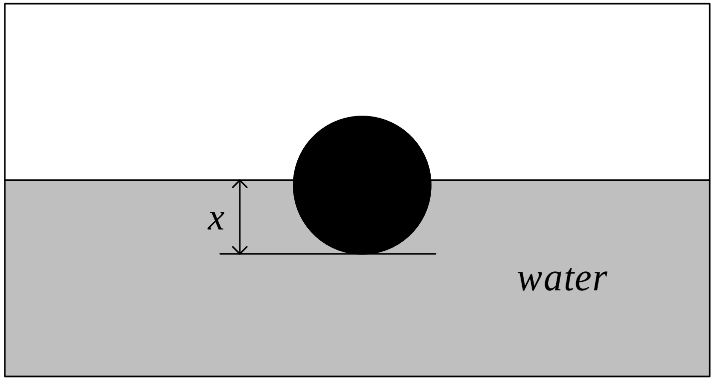


Figure : Floating ball problem

Let us assume $x_L = 0,\ x_U = 0.11$

$$f(x_L) = f(0) = (0)^3 - 0.165(0)^2 + 3.993\times10^{-4} = 3.993\times10^{-4}$$

$$f(x_U) = f(0.11) = (0.11)^3 - 0.165(0.11)^2 + 3.993\times10^{-4} = -2.662\times10^{-4}$$

Hence, $f(x_L)f(x_U) = f(0)f(0.11) = \left(3.993\times10^{-4}\right)\left(-2.662\times10^{-4}\right) < 0$

Iteration 1

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

$$= \frac{0.11\times3.993\times10^{-4} - 0\times\left(-2.662\times10^{-4}\right)}{3.993\times10^{-4} - \left(-2.662\times10^{-4}\right)}$$

$$= 0.0660$$

$$f(x_r) = f(0.0660) = (0.0660)^3 - 0.165(0.0660)^2 + \left(3.993\times10^{-4}\right)$$

$$= -3.1944\times10^{-5}$$

$$f(x_L)f(x_r) = f(0)f(0.0660) = (+)(-) < 0$$

Hence, $x_L = 0,\ x_U = 0.0660$

Iteration 2

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

$$= \frac{0.0660 \times 3.993 \times 10^{-4} - 0 \times (-3.1944 \times 10^{-5})}{3.993 \times 10^{-4} - (-3.1944 \times 10^{-5})}$$

$$= 0.0611$$

$$f(x_r) = f(0.0611) = (0.0611)^3 - 0.165(0.0611)^2 + (3.993 \times 10^{-4})$$

$$= 1.1320 \times 10^{-5}$$

$$f(x_L) f(x_r) = f(0) f(0.0611) = (+)(+) > 0$$

Hence, $x_L = 0.0611,\ x_U = 0.0660$

$$\in_a = \left| \frac{0.0611 - 0.0660}{0.0611} \right| \times 100 \cong 8\%$$

Iteration 3

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

$$x_r = \frac{0.0660 \times 1.132 \times 10^{-5} - 0.0611 \times \left(-3.1944 \times 10^{-5}\right)}{1.132 \times 10^{-5} - \left(-3.1944 \times 10^{-5}\right)}$$

$$= 0.0624$$

$$f\left(x_r\right) = -1.1313 \times 10^{-7}$$

$$f\left(x_L\right) f\left(x_r\right) = f\left(0.0611\right) f\left(0.0624\right) = (+)(-) < 0$$

Hence, $x_L = 0.0611,\ x_U = 0.0624$

$$\in_a = \left| \frac{0.0624 - 0.0611}{0.0624} \right| \times 100 \cong 2.05\%$$

$$\left| \in_a \right| \le 0.5 \times 10^{2-m} \Rightarrow 0.02 \le 0.5 \times 10^{2-m}$$

$$0.04 \le 10^{2-m} \Rightarrow \log(0.04) \le 2 - m$$

$$m \le 2 - \log(0.04) \Rightarrow m \le 2 - (-1.3979)$$

$$m \le 3.3979 \Rightarrow So,\ m = 3$$

From above, giving that *m=3*:

So, the number of significant digits at least correct in the estimated root of 0.062377619 at the end of 4th iteration is 3.

Iteration	x_L	x_U	x_r	$\lvert\in_a\rvert\%$	$f(x_r)$
1	0.0000	0.1100	0.0660	N/A	$-3.1944\text{x}10^{-5}$
2	0.0000	0.0660	0.0611	8.00	$1.1320\text{x}10^{-5}$
3	0.0611	0.0660	0.0624	2.05	$-1.1313\text{x}10^{-7}$
4	0.0611	0.0624	0.062377619	0.02	$-3.3471\text{x}10^{-10}$

Table : Root of $f(x) = x^3 - 0.165x^2 + 3.993 \times 10^{-4} = 0$ for False-Position Method.

Hand Calculation Example

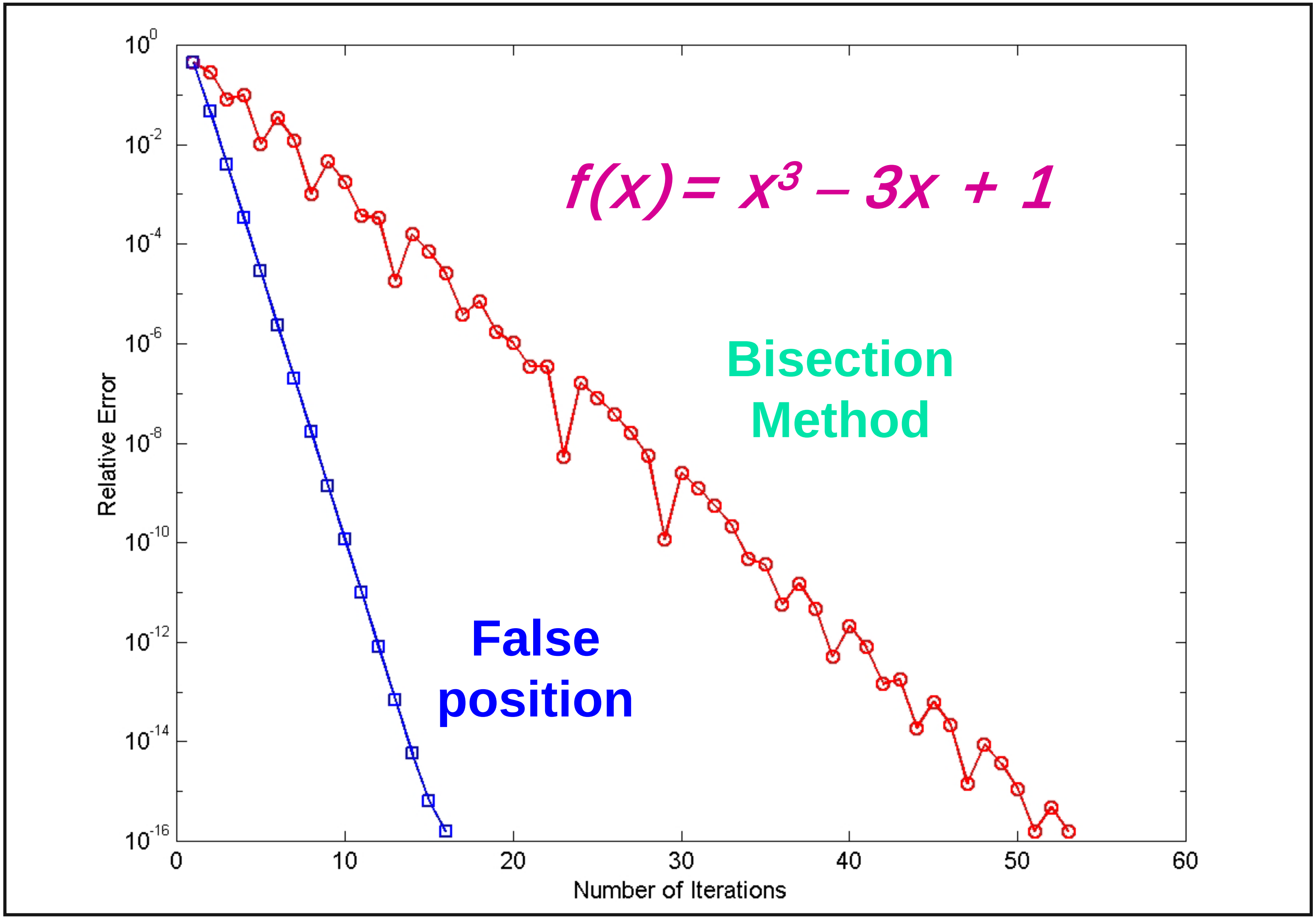
False-Position

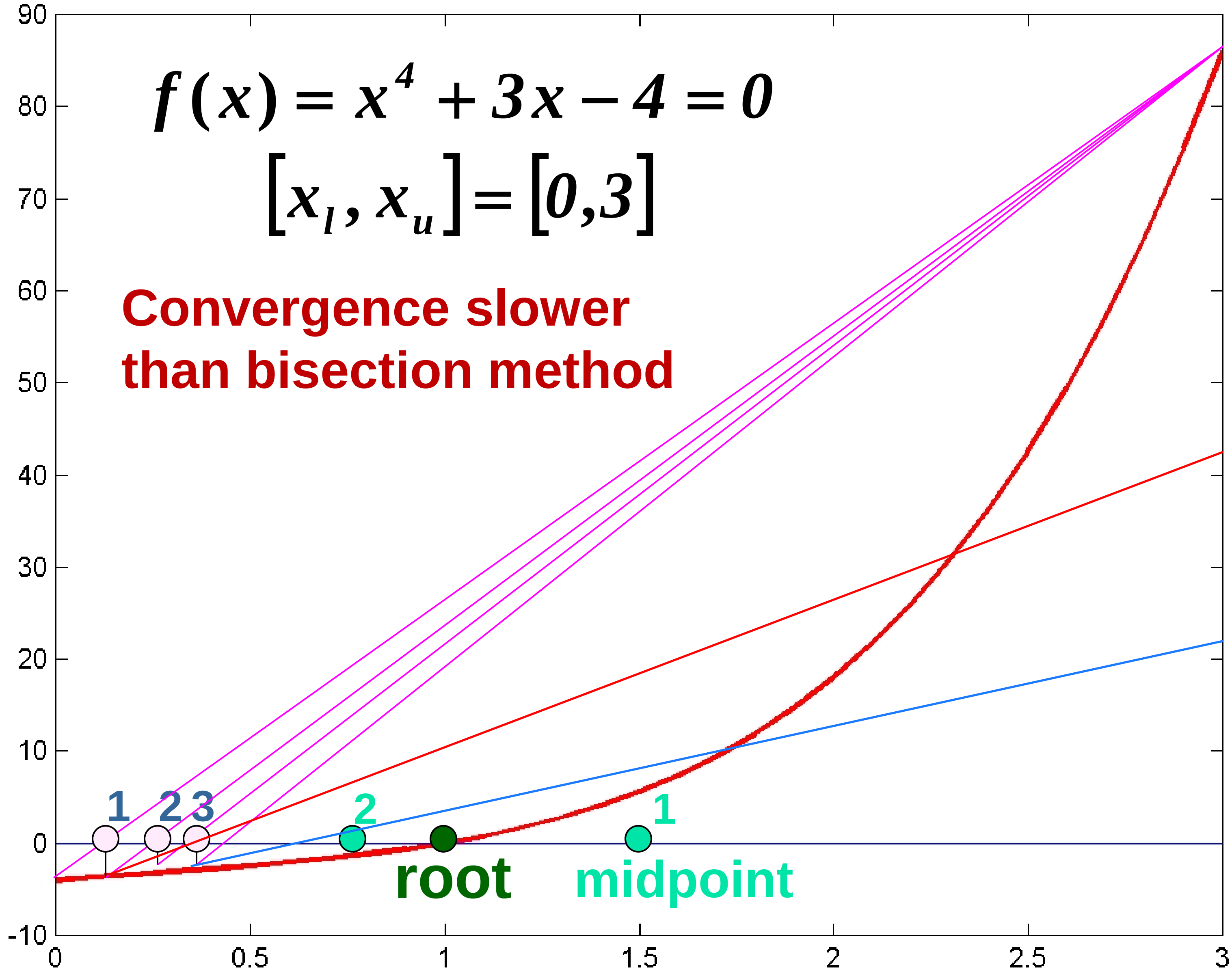
$$Example:\ f(x) = x^2 - 2x - 3 = 0$$

$$initial\ estimeates\ [x_l, x_u] = [2.0,\ 3.2]$$

$iter$	x_l	x_u	x_r	$f(x_r)$
1	2.0	3.2	2.9375	-0.2461
2	2.9375	3.2	2.9968	-0.01207
3	2.99698	3.2	2.999856	-0.000576
4	2.999856	3.2	2.99999315	-0.0000274

Rate of Convergence

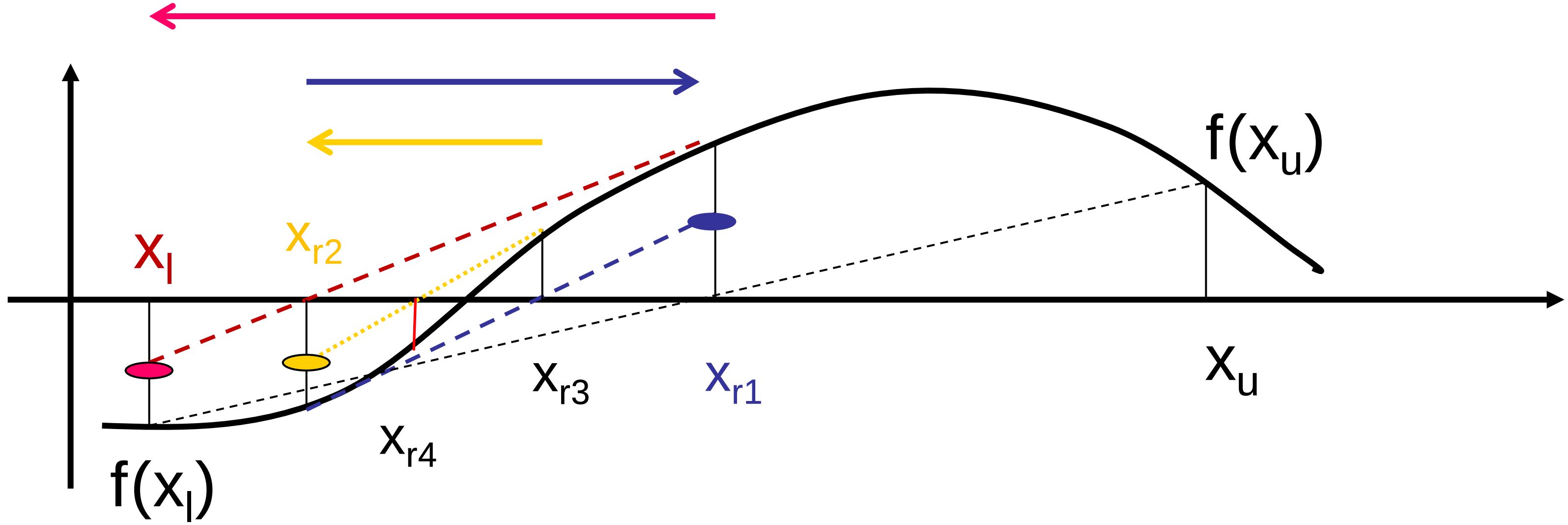




A hard problem

Modified Regula Falsi

- Because *Regula Falsi* can be fatally slow some of the time (which is too often)
- **Trick:** drop the line from f_l or f_u to some fraction of its height, artificially change slope to cut off more of other side
- The root will flip between left and right interval
- If the root is in the **left segment [x_l, x_r]**, draw line between **(x_l, $f(x_l)*0.5$)** *and* **(x_r, $f(x_r)$)**
- Else (in the **right segment [x_r, x_u])**, draw line between **(x_r, $f(x_r)$)** *and* **(x_u, $f(x_u)*0.5$)**



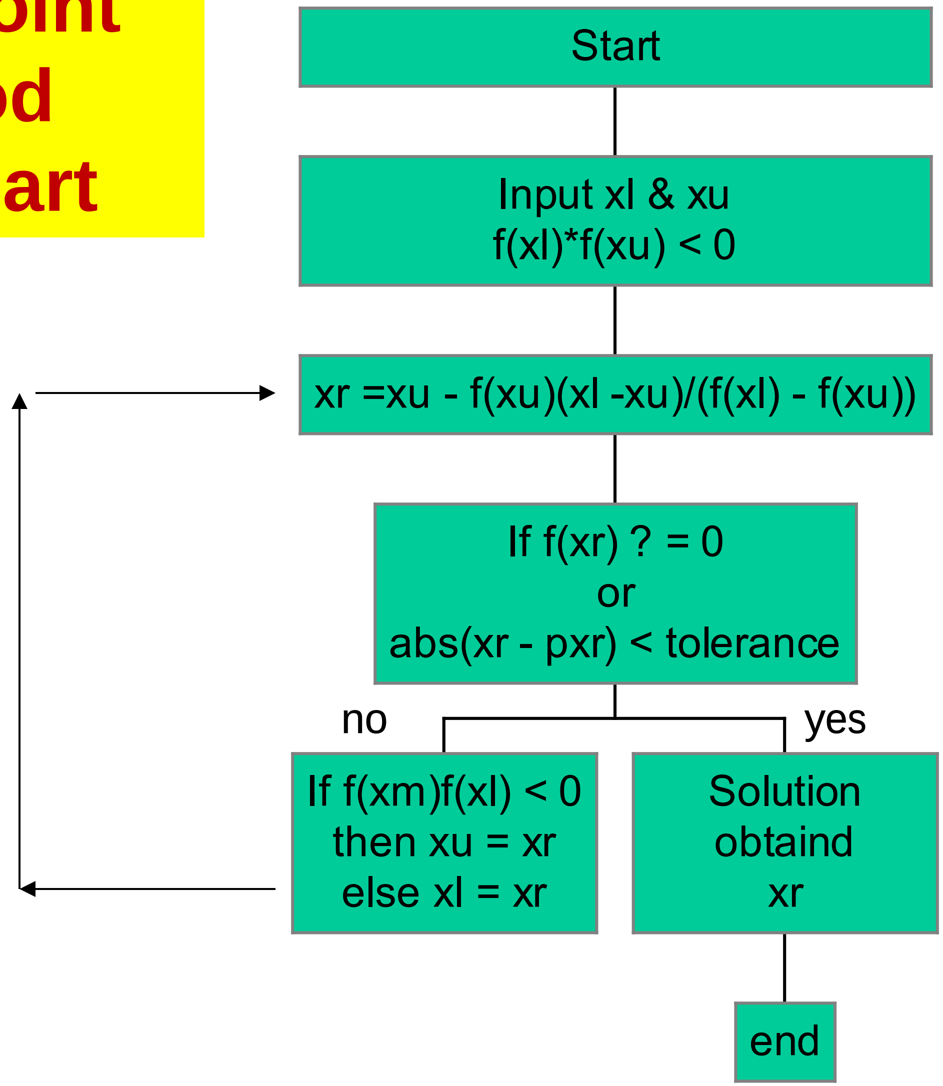
$$x_{r1} = \frac{x_U f(x_L) - x_L f(x_U)}{f(x_L) - f(x_U)}$$

$$x_{r2} = \frac{x_{r1} f(x_L)/2 - x_L f(x_{r1})}{f(x_L)/2 - f(x_{r1})}$$

$$x_{r3} = \frac{x_{r1} f(x_{r2}) - x_{r2} f(x_{r1})/2}{f(x_{r2}) - f(x_{r1})/2}$$

$$x_{r4} = \frac{x_{r3} f(x_{r2})/2 - x_{r2} f(x_{r3})}{f(x_{r2})/2 - f(x_{r3})}$$

False-point Method Flowchart



False-position (Regula-Falsi)

Linear Interpolation Method

```
function [x,y] = false_position(func)

% Find root near x1 using the false position method.
% Input:   func       string containing name of function
%          xl,xu      initial guesses
%          es         allowable tolerance in computed root
%          maxit      maximum number of iterations
% Output:  x          row vector of approximations to root

xl = input('enter lower bound xl = ');
xu = input('enter upper bound xu = ');
es = input('allowable tolerance es = ');
maxit = input('maximum number of iterations maxit = ');

a(1) = xl; b(1) = xu;
ya(1) = feval(func, a(1)); yb(1)=feval(func, b(1));
if ya(1) * yb(1) > 0.0
     error('Function has same sign at end points')
end
for i = 1:maxit
     x(i) = b(i) - yb(i)*(b(i)-a(i))/(yb(i)-ya(i));
     y(i) = feval(func, x(i));
     if y(i) == 0.0
        disp('exact zero found'); break;
     elseif y(i) * ya(i) < 0
        a(i+1) = a(i); ya(i+1) = ya(i);
        b(i+1) = x(i); yb(i+1) = y(i);
     else
        a(i+1) = x(i); ya(i+1) = y(i);
        b(i+1) = b(i); yb(i+1) = yb(i);
     end;
      if((i > 1) & (abs(x(i) - x(i-1)) < es))
          disp('False position method has converged'); break
      end
     iter = i;
end
if(iter >= maxit)
        disp('zero not found to desired tolerance');
end
n=length(x); k=1:n; out = [k'  a(1:n)'  b(1:n)'  x'  y'];
disp('      step      xl        xu        xr        f(xr)')
disp(out)
```

Linear interpolation

Exercises

1. Find the real root of the equation till 2 decimal place $x^3 - 2x^2 + 3x = 5$ between the points 1 and 2.

   Result 1.843734

2. Find the real root of the equation till 2 decimal place $\sin x + x - 1 = 0$.

   Result 0.510973

Fixed Point iterative Method

A real root of F(x) = x instead of f(x) = 0 can be found from the method of iteration

EXAMPLE

$$f(x) = \cos x - 4x + 5 = 0$$

It can be written as

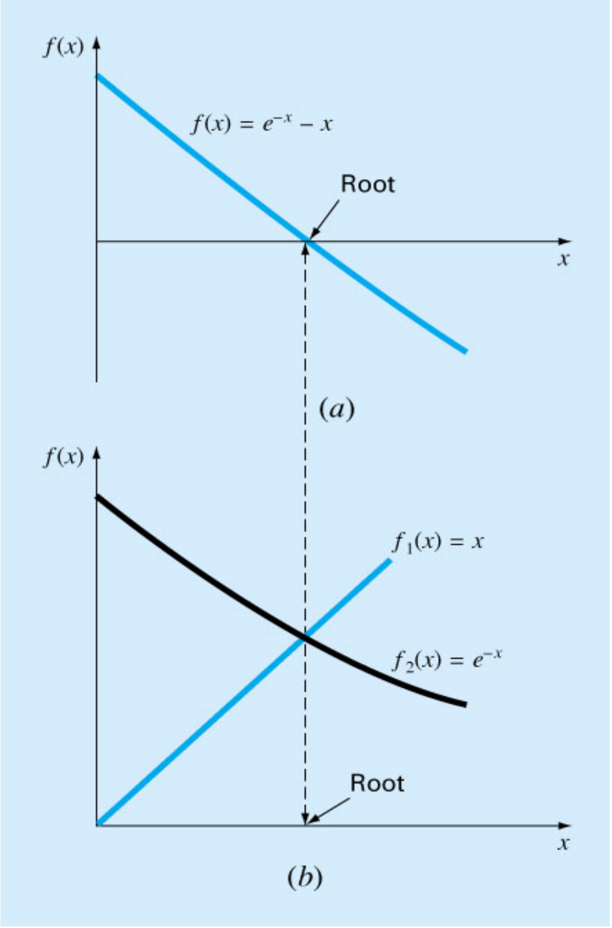
$$x = \frac{1}{4}(\cos x + 5) = F(x)$$

EXAMPLE

$$f(x) = x^2 - x - 2 \qquad x > 0$$

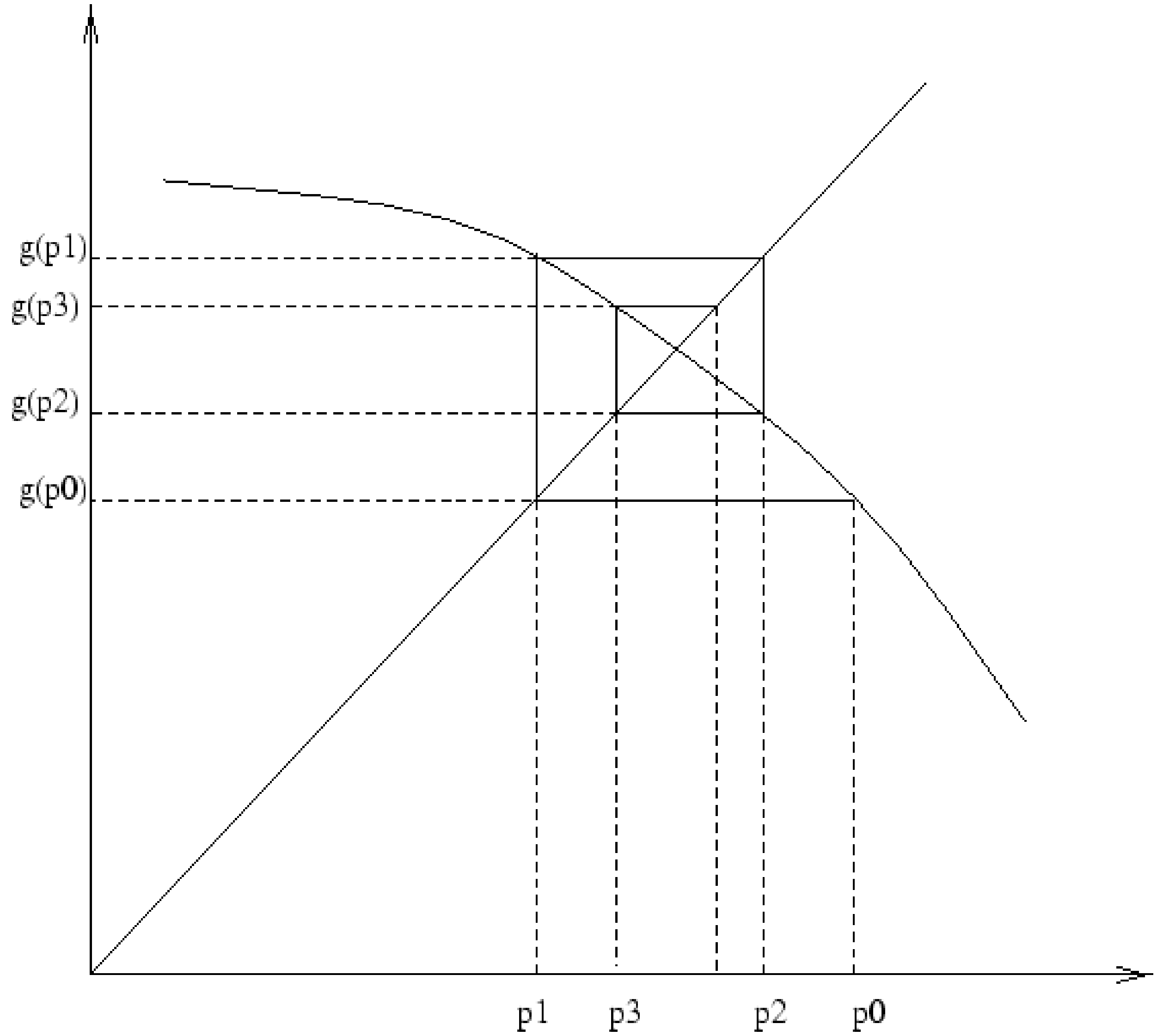
$$g(x) = x^2 - 2 \text{ or } g(x) = \sqrt{x+2} \text{ or } g(x) = 1 + \frac{2}{x} \quad \ldots$$

Which g(x) converges?



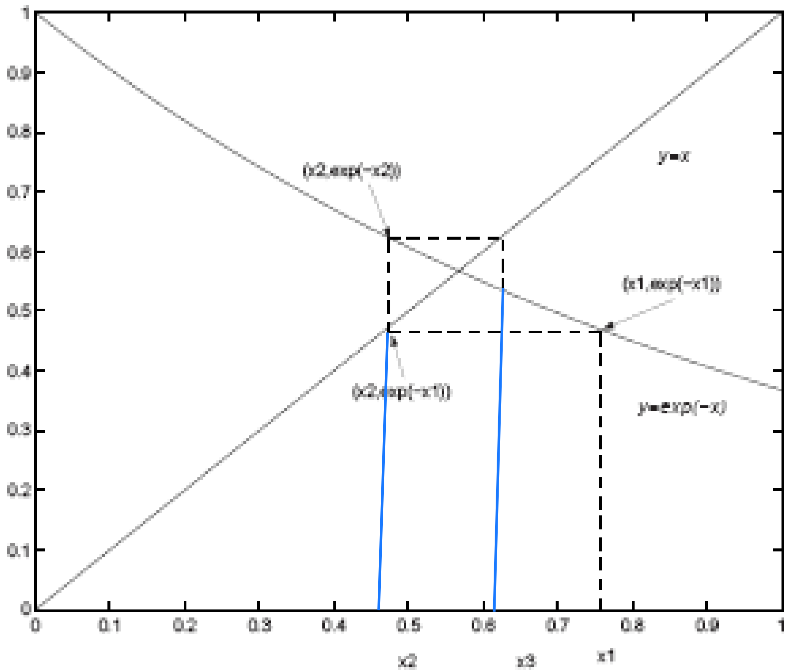
Fixed point iteration

- We get successive iterates as follows:
- Start at the initial x value on the x-axis(p_0)
- Go vertically to the curve.
- Then to the line y = x
- Then to the curve
- Then the line. **Repeat!**



- Consider x which satisfies $x = e^{-x}$
- Can re-write this as $-\log x = x$
- Which one converges?

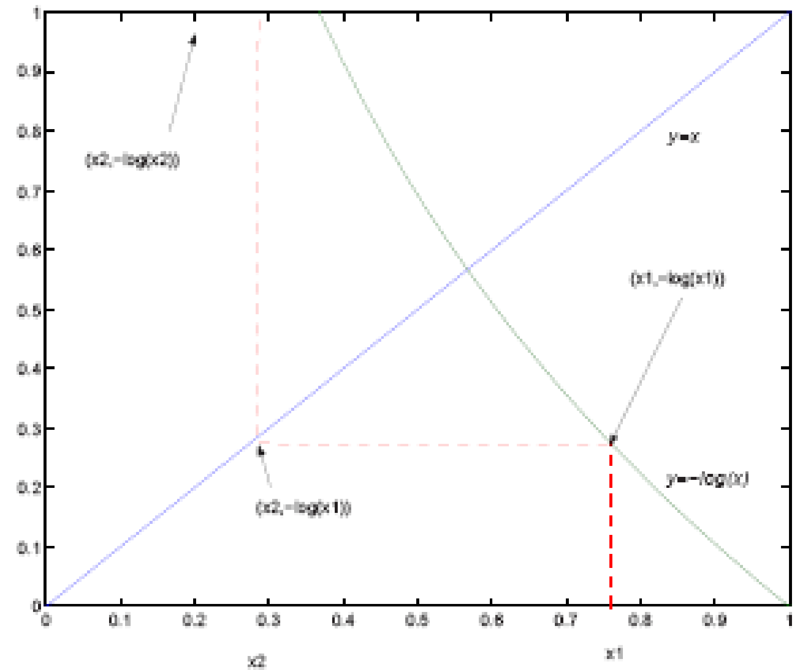
Algorithm 1



Stable

$$x_{n+1} = e^{-x_n}$$

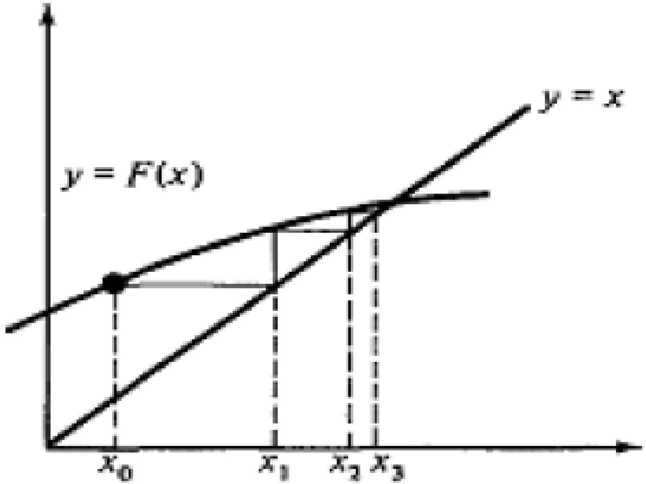
Algorithm 2



Unstable

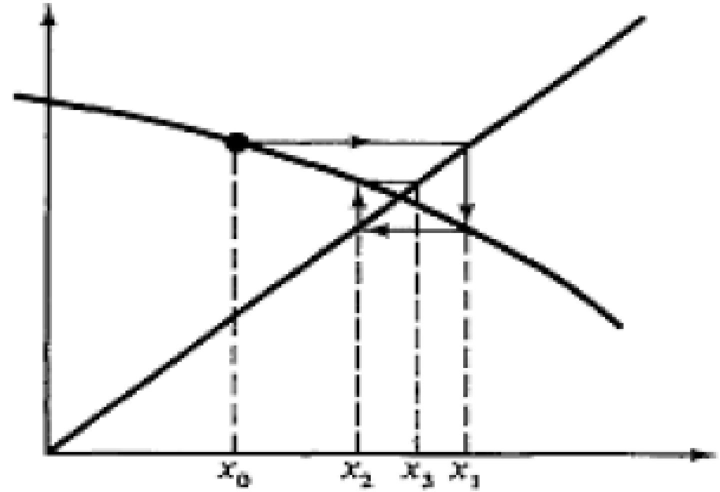
$$x_{n+1} = -\log x_n$$

Four basic convergent conditions of $|F'(x)|$



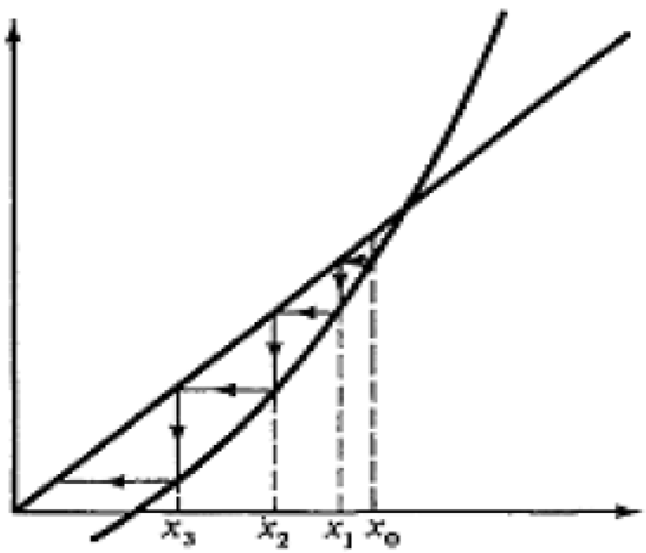
$0 < F'(x) < 1$

Fig. 1.2.1



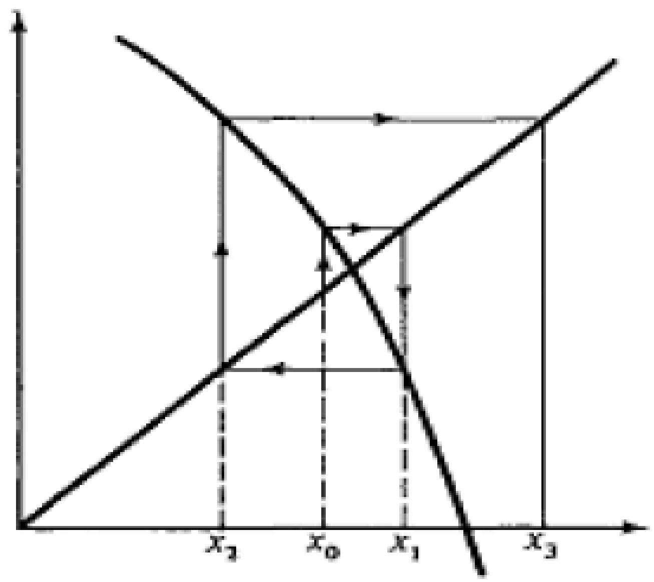
$-1 < F'(x) < 0$

Fig. 1.2.2



$F'(x) > 1$

Fig. 1.2.3



$F'(x) < -1$

Fig. 1.2.4

Convergence Conditions

- Any arbitrary approximation x_0, x_1, x_2 does not assure that it will converge to the actual root x of the equation.
  - E.g. $x = 10^x + 1$, if $x_0 = 0$, $x_1 = 2$, $x_2 = 101$, …. that does not converge to the actual root x. As n increase, x_n increases without limit!
- The equation $\boxed{x = F(x)}$ converges to the real root x,
  - if $F(x)$ is continuous
  - If $| F'(x) | < 1$
- The equation $x = F(x)$ does not converges,
  - if $| F'(x) | > 1$
- Therefore, $f(x) = 0$ is re-written as $x = F(x)$ in such a way that $\mathbf{| F'(x) | < 1}$

Example: $f(x) = x^2 - 2x - 3 \rightarrow x_{1,2} = -1, 3$

- $x=g_1(x)=\sqrt{2x+3}$ Start with $x_0 = 4$
  - $x_1= \sqrt{11}=3.31662$
  - $x_2= \sqrt{9.63325}=3.10375$
  - $x_3= \sqrt{9.20750}=3.03439$
  - $x_4= \sqrt{9.06877}=3.01144$
  - $x_5= \sqrt{9.02288}=3.00381$
  - …converging to 3

- $x=g_2(x)=3/(x-2)$ Start with $x_0 = 4$
  - $x_1= 1.5$
  - $x_2=-6$
  - $x_3=-0.375$
  - $x_4=-1.263158$
  - $x_5= -0.919355$
  - $x_6= -1.02762$
  - $x_7= -0.990876$
  - $x_8= -1.00305$
  - …converging to -1

- $x=g_3(x)=(x^2-3)/2$ Start with $x_0 = 4$
  - $x_1= 6.5$
  - $x_2=19.625$
  - $x_3=191.070$
  - …seems diverging

Example: Solve $x = 2 + sin(x)/2$

Solution

Here $f(x) = 2 + sin(x)/2$

Starting with $x_0 = 2$
we calculate x_1, x_2,...

x_0	**2**
$x_1 = f(x_0)$	**2.454648713**
$x_2 = f(x_1)$	**2.31708862**
$x_3 = f(x_2)$	**2.367105575**
$x_4 = f(x_3)$	**2.349674771**
$x_5 = f(x_4)$	**2.355850929**
$x_6 = f(x_5)$	**2.353674837**
$x_7 = f(x_6)$	**2.354443099**
$x_8 = f(x_7)$	**2.354172058**
$x_9 = f(x_8)$	**2.354267705**
$x_{10} = f(x_9)$	**2.354233955**
$x_{11} = f(x_{10})$	**2.354245864**

Example: Find the real root of the equation

$$g(x) = x^3 + x^2 - 1 = 0$$

Rewrite $g(x)$

$$x^3 + x^2 - 1 = 0$$

or, $x^3 + x^2 = 1$

or, $x^2(x+1) = 1$

or, $x^2 = 1/(x+1)$

or, $x = 1/\sqrt{(x+1)}$

Let, $x_0 = 0.75$

$x_0 = 0.7500000$
$x_1 = 0.7559289$
$x_2 = 0.7546517$
$x_3 = 0.7549263$
$x_4 = 0.7548672$
$x_5 = 0.7548799$
$x_6 = 0.7548772$
$x_7 = 0.7548778$
$x_8 = 0.7548776$
$x_9 = 0.7548777$
$x_{10} = 0.7548777$

Example: Compute zeros of $f(x) = e^x - 4 - 2x$

Scheme 1:

$$x = g(x) = \frac{1}{2}(e^x - 4)$$

$$x_{n+1} = \frac{1}{2}(e^{x_n} - 4)$$

$$g'(x) = \frac{1}{2}e^x$$

with $|g'(x)| < 1$ for $-\infty < x < 0.693$

Let us choose $x_0 = -2,$

$$\Rightarrow \quad x_1 = \frac{1}{2}(e^{-2} - 4) = -1.9323$$

$$x_2 = \frac{1}{2}(e^{-1.9323} - 4) = -1.9276$$

$$x_3 = -1.9273, \; x_4 = -1.9272,$$

$$x_5 = -1.9272$$

Scheme 2:

$$x = g(x) = \ln(4 + 2x)$$

$$x_{n+1} = \ln(4 + 2x_n), \qquad x > -2$$

$$g'(x) = \frac{2}{4 + 2x} \quad \Rightarrow$$

$$\cancel{|g'(x)| < 1, \; x \in (-\infty, -3)}$$

$$|g'(x)| < 1, \quad x \in (-1, \infty)$$

Let $x_0 = 0, \quad \Rightarrow$

$$x_1 = 1.3863, \; x_2 = 1.9129,$$

$$x_3 = 2.0574, \; x_4 = 2.0937,$$

$$x_5 = 2.1026, \; x_6 = 2.1048,$$

$$x_7 = 2.1053, \; x_8 = 2.1054,$$

$$x_9 = 2.1054$$

Example:

- Solve $f(x) = 0$ by rewriting as $x = g(x)$ and iterating $x_{n+1} = g(x_n)$
- $f(x) = x^3 - \sin(x) = 0$ (has 3 roots)

$$x = \sqrt[3]{\sin(x)}$$
$$x_{n+1} = \sqrt[3]{\sin(x_n)}$$

n	x_n
1	1.0
2	0.944
3	0.932
4	0.929
5	0.929

Converges

$$x = \frac{\sin x}{x^2}$$
$$x_{n+1} = \frac{\sin x_n}{x_n^2}$$

n	x_n
1	1.0
2	0.841
3	1.053
4	0.783
5	1.149

Diverges

Drawbacks

- We need an *approximate initial guesses* x_0.
- It is also a *slower* method to find the root.
- If the equation has more than one roots, then this method can find *only* one of them.

Exercise: Find the real root of the equation using iterative method (till 2 decimal places).

$$e^{-x} = 10x$$

Answer: 0.091276527

Newton-Raphson Method

$$\Delta x = \frac{-f(x_1)}{m} = \frac{-f(x_1)}{f'(x_1)}$$

$$x^{k+1} = x_1^{\,k} + \Delta x,\ etc.$$

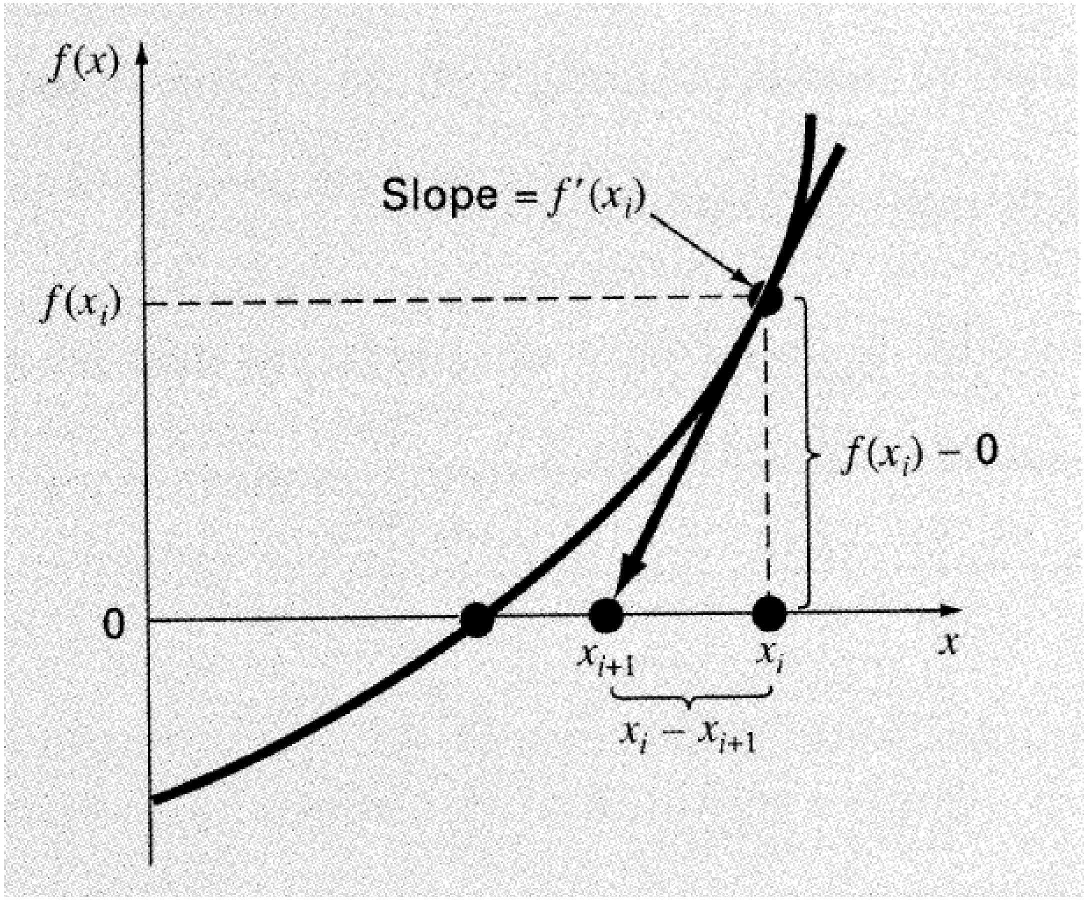
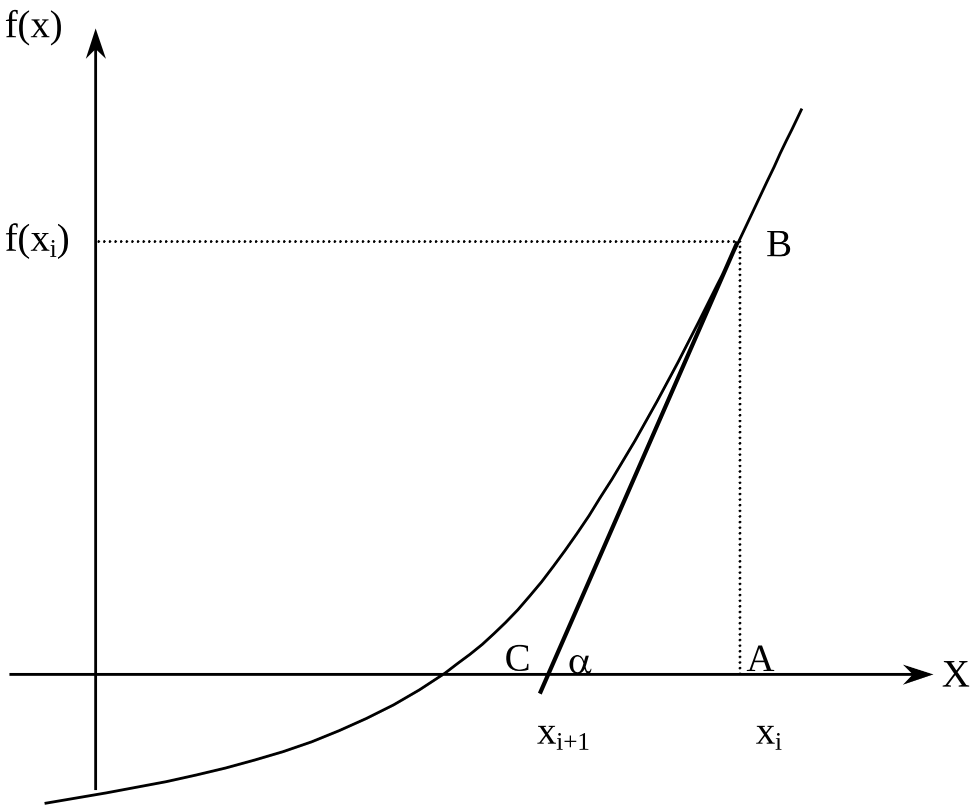


Figure 1 Geometrical illustration of the Newton-Raphson method.

Derivation



$$\tan(\alpha) = \frac{AB}{CA}$$

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

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

Figure 2 Derivation of the Newton-Raphson method.

Algorithm for Newton-Raphson Method

Step 1 Evaluate $f'(x)$ symbolically.

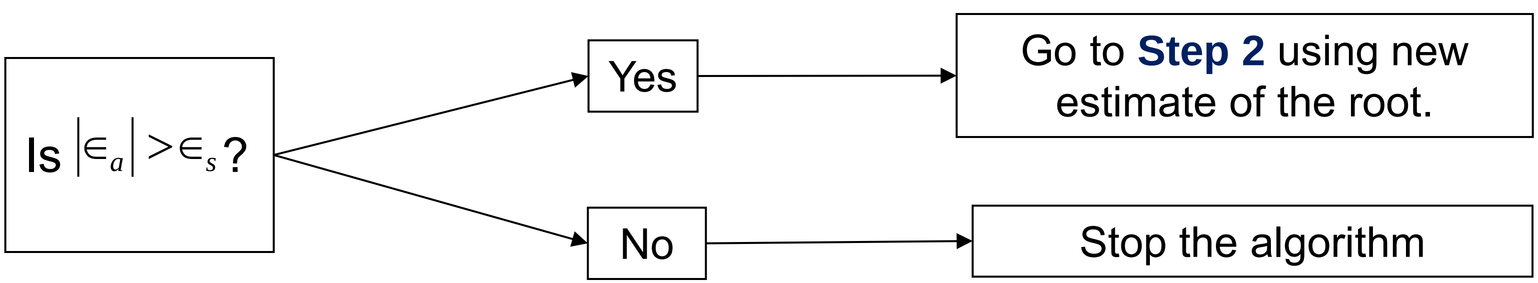
Step 2 Use an initial guess of the root x_i to estimate the new value of the root x_{i+1} as

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

Step 3 Find the absolute relative approximate error $|\in_a|$ as

$$|\in_a| = \left|\frac{x_{i+1} - x_i}{x_{i+1}}\right| \times 100$$

Step 4 Compare the absolute relative approximate error with the pre-specified relative error tolerance $\in_s$.



Also, check if the number of iterations has exceeded the maximum number of iterations allowed. If so, one needs to terminate the algorithm and notify the user.

Example 1 The equation that gives the depth *x* in meters to which the ball is submerged under water is given by

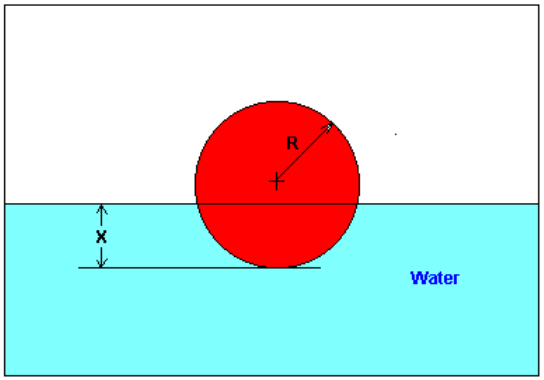


Figure 3 Floating ball problem.

$$f(x) = x^3 - 0.165x^2 + 3.993 \times 10^{-4}$$

Use the Newton's method of finding roots of equations to find

a) The depth 'x' to which the ball is submerged under water. Conduct three iterations to estimate the root of the above equation.

b) The absolute relative approximate error at the end of each iteration

c) The number of significant digits at least correct at the end of each iteration.

Solution To understand how this method works to find the root of an equation, the graph of f(x) is shown in the Figure

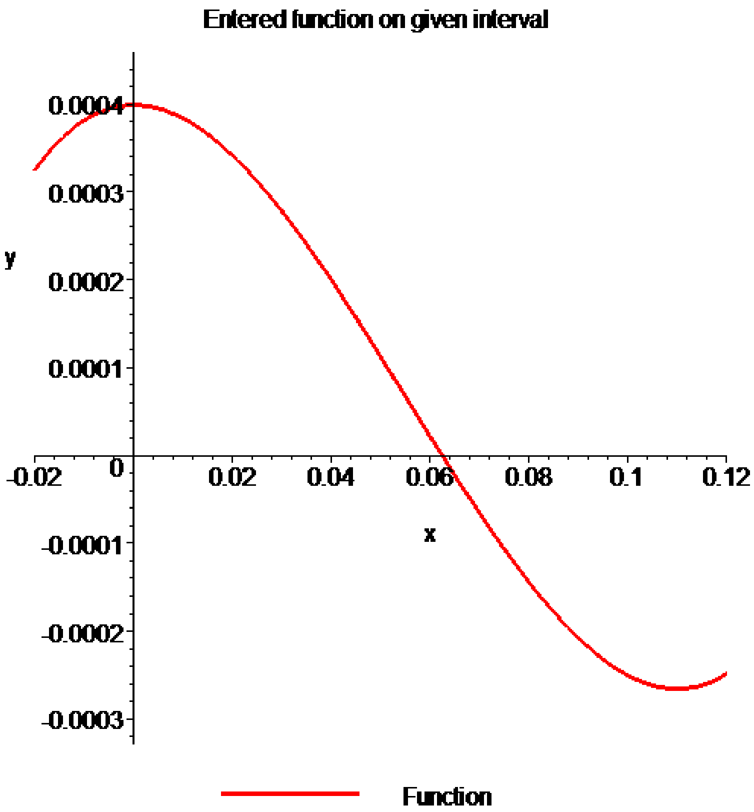


Figure 4 Graph of the function f(x)

Solve for $f'(x)$

$$f(x) = x^3 - 0.165x^2 + 3.993\times10^{-4}$$

$$f'(x) = 3x^2 - 0.33x$$

Let us assume the initial guess of the root of $f(x)=0$ is $x_0 = 0.05\text{m}$. This is a reasonable guess (discuss why $x=0$ and $x=0.11\text{m}$ are not good choices) as the extreme values of the depth x would be 0 and the diameter (0.11 m) of the ball.

Iteration 1 The estimate of the root is

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

$$= 0.05 - \frac{(0.05)^3 - 0.165(0.05)^2 + 3.993\times10^{-4}}{3(0.05)^2 - 0.33(0.05)}$$

$$x_1 = 0.05 - \frac{1.118\times10^{-4}}{-9\times10^{-3}}$$
$$= 0.05 - (-0.01242)$$
$$= 0.06242$$

The absolute relative approximate error $|\in_a|$ at the end of Iteration 1 is

$$|\in_a| = \left|\frac{x_1 - x_0}{x_1}\right| \times 100$$
$$= \left|\frac{0.06242 - 0.05}{0.06242}\right| \times 100$$
$$= 19.90\%$$

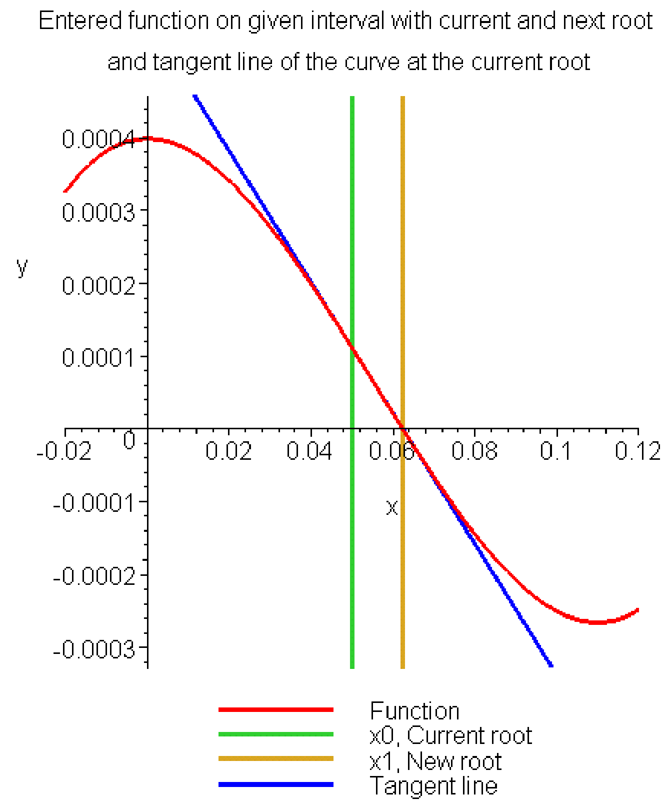


Figure 5 Estimate of the root for the first iteration.

The number of significant digits at least correct is 0, as you need an absolute relative approximate error of 5% or less for at least one significant digits to be correct in your result.

Iteration 2 The estimate of the root is $x_2 = x_1 - \dfrac{f(x_1)}{f'(x_1)}$

$$x_2 = 0.06242 - \frac{(0.06242)^3 - 0.165(0.06242)^2 + 3.993\times10^{-4}}{3(0.06242)^2 - 0.33(0.06242)}$$

$$= 0.06242 - \frac{-3.97781\times10^{-7}}{-8.90973\times10^{-3}} = 0.06242 - \left(4.4646\times10^{-5}\right)$$

$$= 0.06238$$

The absolute relative approximate error $|\in_a|$ at the end of Iteration 2 is

$$|\in_a| = \left|\frac{x_2 - x_1}{x_2}\right| \times 100$$

$$= \left|\frac{0.06238 - 0.06242}{0.06238}\right| \times 100$$

$$= 0.0716\%$$

The maximum value of *m* for which $|\in_a| \le 0.5\times10^{2-m}$ is 2.844. Hence, the number of significant digits at least correct in the answer is 2.

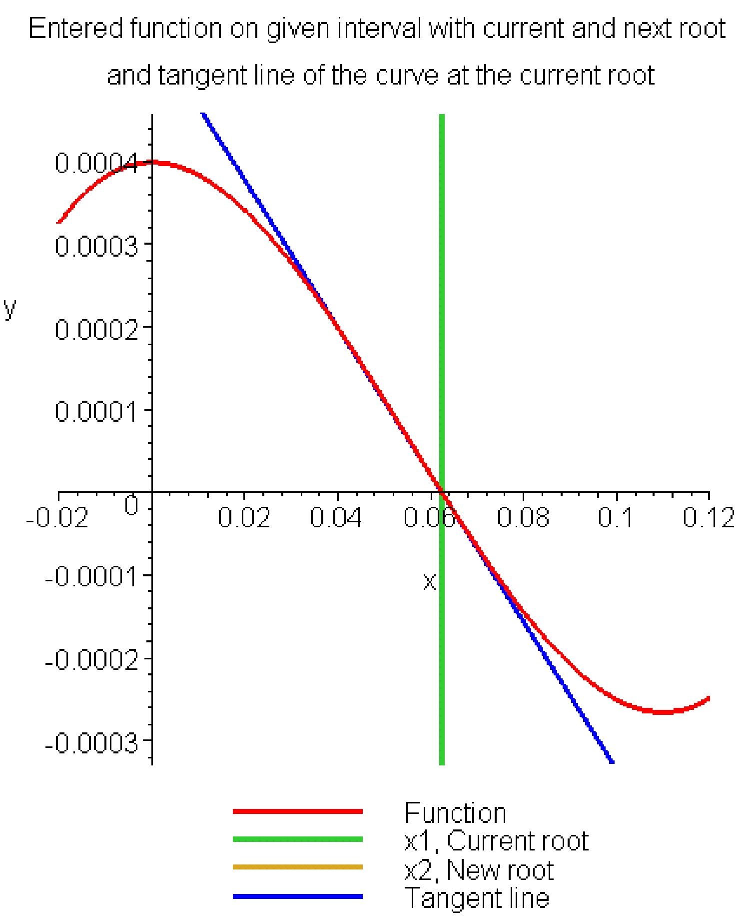


Figure 6 Estimate of the root for the Iteration 2.

Iteration 3 The estimate of the root is $x_3 = x_2 - \dfrac{f(x_2)}{f'(x_2)}$

$$x_3 = 0.06238 - \frac{(0.06238)^3 - 0.165(0.06238)^2 + 3.993\times10^{-4}}{3(0.06238)^2 - 0.33(0.06238)}$$

$$= 0.06238 - \frac{4.44\times10^{-11}}{-8.91171\times10^{-3}}$$

$$= 0.06238 - \left(-4.9822\times10^{-9}\right)$$

$$= 0.06238$$

The absolute relative approximate error $|\in_a|$ at the end of Iteration 3 is

$$|\in_a| = \left|\frac{x_2 - x_1}{x_2}\right|\times100$$

$$= \left|\frac{0.06238 - 0.06238}{0.06238}\right|\times100$$

$$= 0\%$$

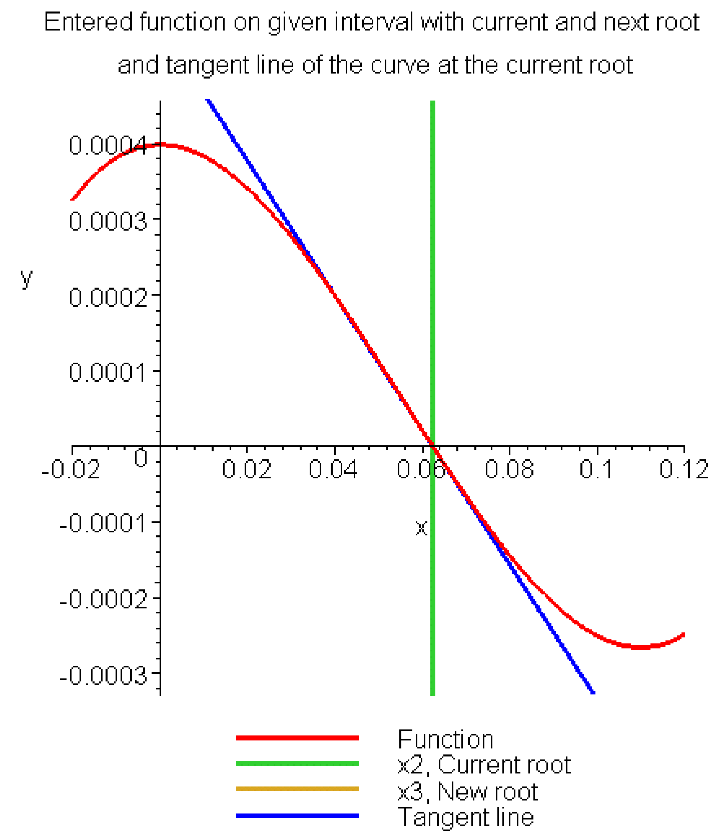
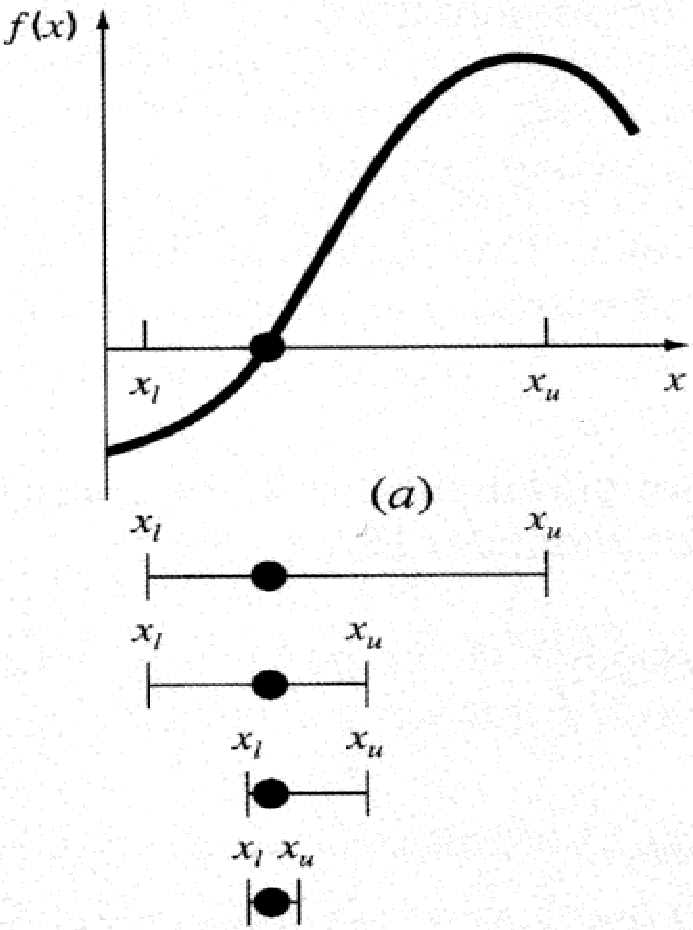


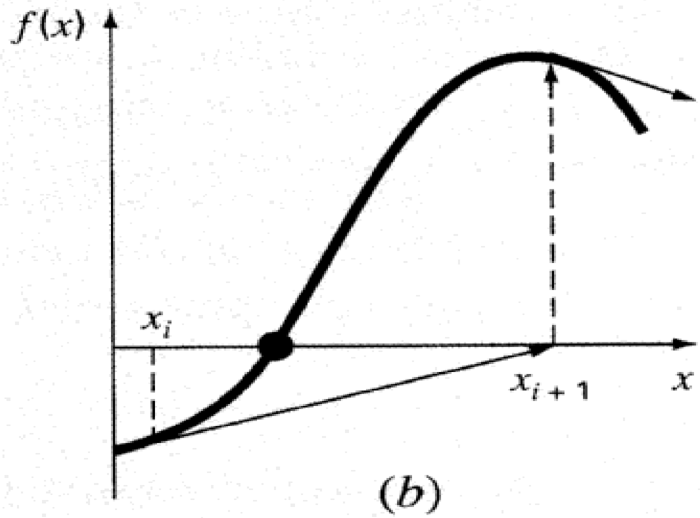
Figure 7 Estimate of the root for the Iteration 3.

The number of significant digits at least correct is 4, as only 4 significant digits are carried through all the calculations.

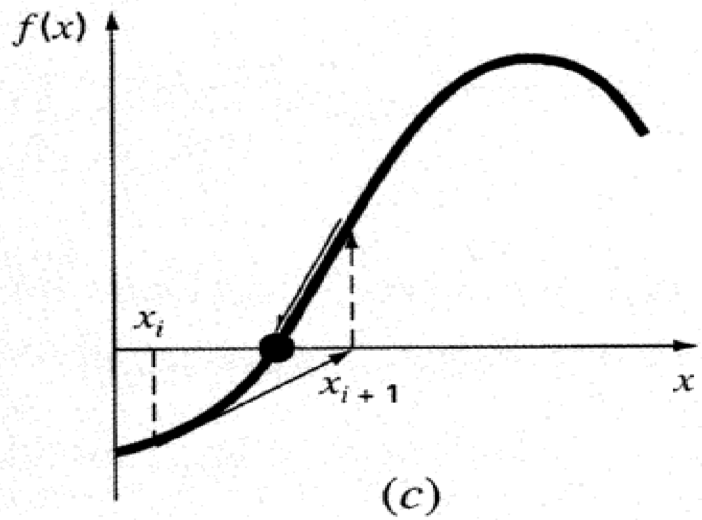
Advantages and Drawbacks



Bisection is always safe



Newton unfavorable



Newton favorable

Advantages:

- Converges fast (quadratic convergence), if it converges.
- Requires only one guess

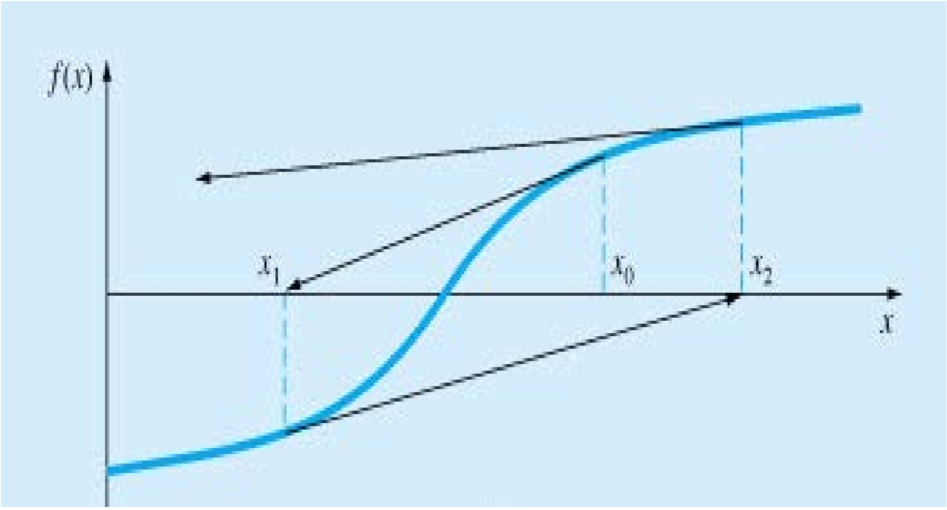
Drawbacks:

- The Newton-Raphson method requires the calculation of the *derivative* of a function, which is not always easy.
- If *F'* vanishes at an iteration point, then the method will fail to converge.
- The method converges to the real root *x*, if

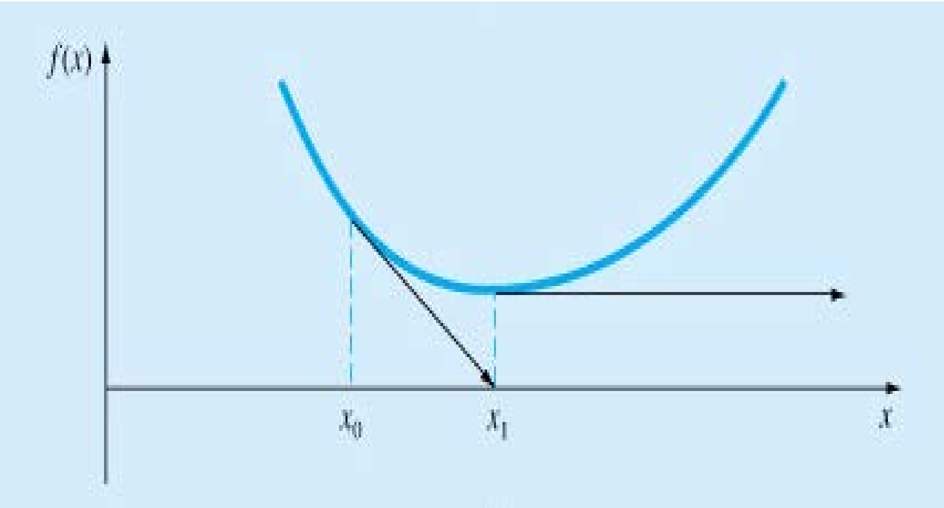
$$\boxed{\left|\frac{f(x)f''(x)}{[f'(x)]^2}\right| < 1} \quad \text{or} \quad \boxed{|f(x)f''(x)| < [f'(x)]^2}$$

- When the step is too large or the value is oscillating, other more conservative methods should take over the case.

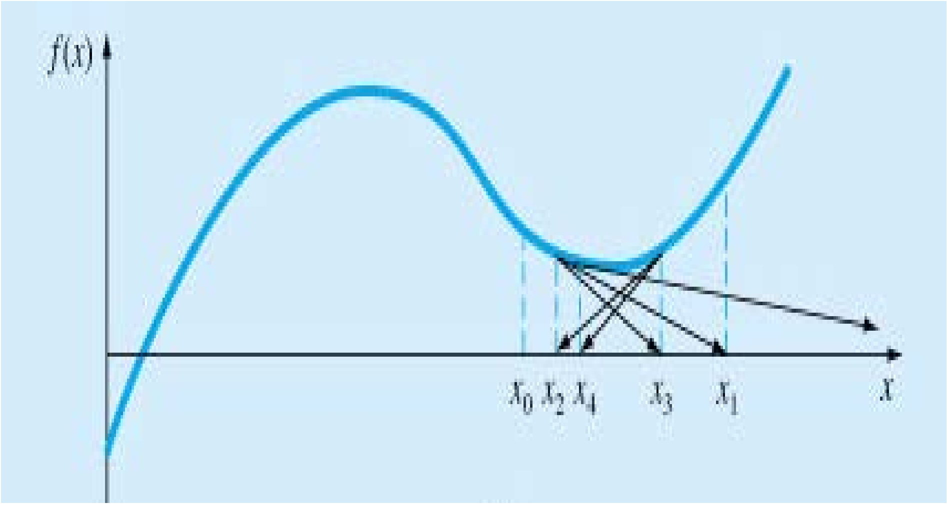
Problems with Newton method



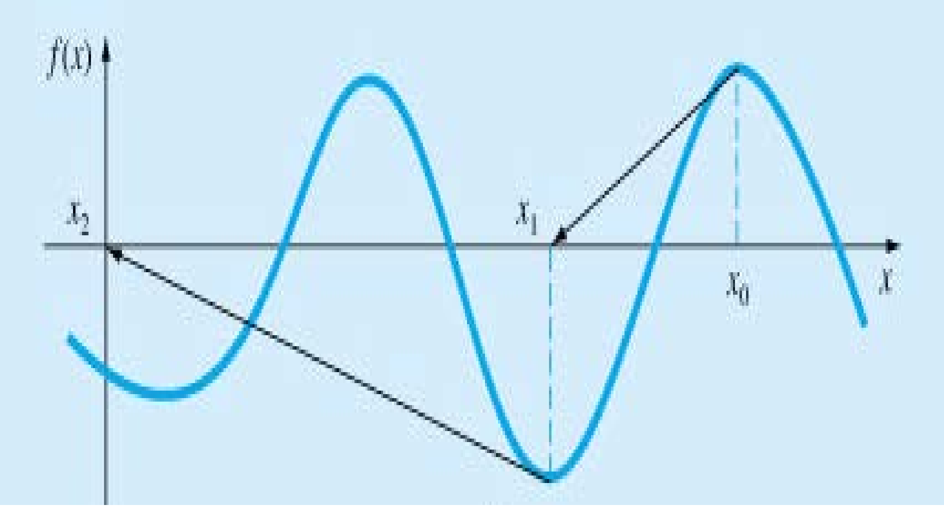
Divergence near inflection point



Division by zero



Oscillations near local maxima or minima



Root Jumping

Problems with Newton method

1. **Divergence at inflection points**

   Selection of the initial guess or an iteration value of the root that is close to the inflection point of the function $f(x)$ may start diverging away from the root in Newton-Raphson method.

For example, to find the root of the equation $f(x)=(x-1)^3+0.512=0$

The Newton-Raphson method reduces to $x_{i+1}=x_i-\dfrac{(x_i^3-1)^3+0.512}{3(x_i-1)^2}$.

Table 1 shows the iterated values of the root of the equation.

The root starts to diverge at Iteration 6 because the previous estimate of 0.92589 is close to the inflection point of $x=1$.

Eventually after 12 more iterations the root converges to the exact value of $x=0.2$.

Iteration Number	x_i
0	5.0000
1	3.6560
2	2.7465
3	2.1084
4	1.6000
5	0.92589
6	−30.119
7	−19.746
18	0.2000

Table 1 Divergence near inflection point.

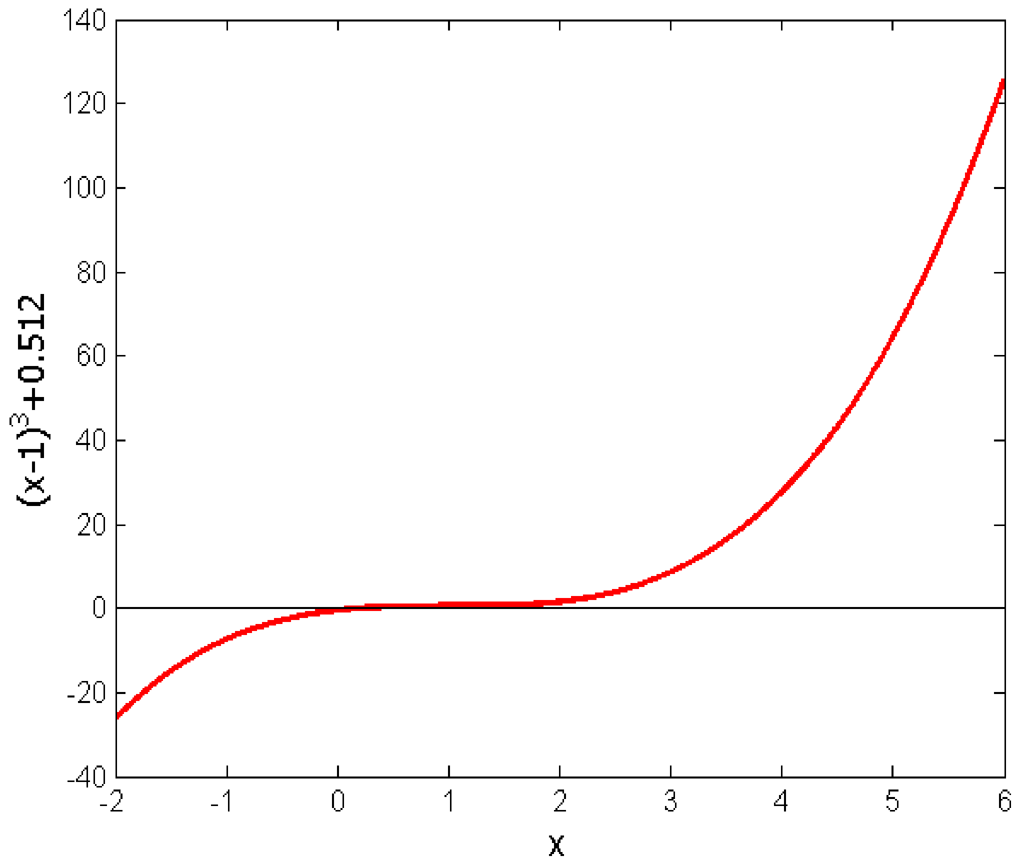


Figure 8 Divergence at inflection point for

$$f(x) = (x-1)^3 + 0.512 = 0$$

2. Division by zero

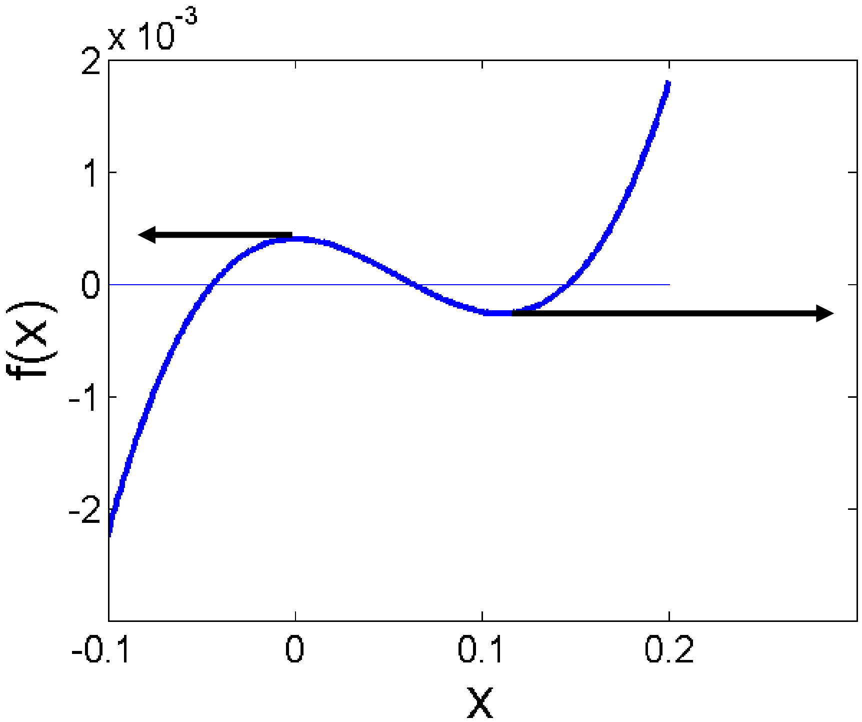
For the equation $f(x) = x^3 - 0.03x^2 + 2.4\times10^{-6} = 0$

the Newton-Raphson method reduces to

$$x_{i+1} = x_i - \frac{x_i^3 - 0.03x_i^2 + 2.4\times10^{-6}}{3x_i^2 - 0.06x_i}$$

For $x_0 = 0$ or $x_0 = 0.02$, the denominator will equal zero.

Figure 9 Pitfall of division by zero or near a zero number



3. Oscillations near local maximum and minimum

Results obtained from the Newton-Raphson method may oscillate about the local maximum or minimum without converging on a root but converging on the local maximum or minimum.

Eventually, it may lead to division by a number close to zero and may diverge.

For example for $f(x) = x^2 + 2 = 0$ the equation has no real roots.

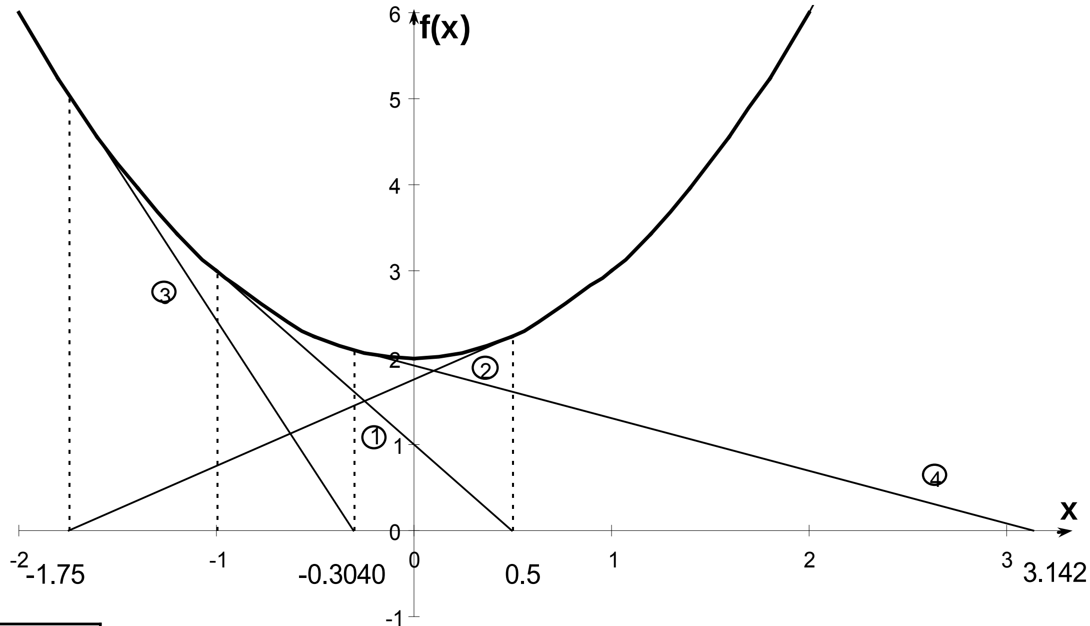


Figure 10 Oscillations around local minima for $f(x) = x^2 + 2$

Table 2 Oscillations near local maxima and mimima in Newton-Raphson method.

Iteration Number	x_i	$f(x_i)$	$\left\|\in_a\right\|\%$
0	–1.0000	3.00	—
1	0.5	2.25	300.00
2	–1.75	5.063	128.571
3	–0.30357	2.092	476.47
4	3.1423	11.874	109.66
5	1.2529	3.570	150.80
6	–0.17166	2.029	829.88
7	5.7395	34.942	102.99
8	2.6955	9.266	112.93
9	0.97678	2.954	175.96

4. Root Jumping

In some cases where the function $f(x)$ is oscillating and has a number of roots, one may choose an initial guess close to a root. However, the guesses may jump and converge to some other root.

For example $f(x) = \sin x = 0$

Choose $x_0 = 2.4\pi = 7.539822$

It will converge to $x = 0$ instead of $x = 2\pi = 6.2831853$

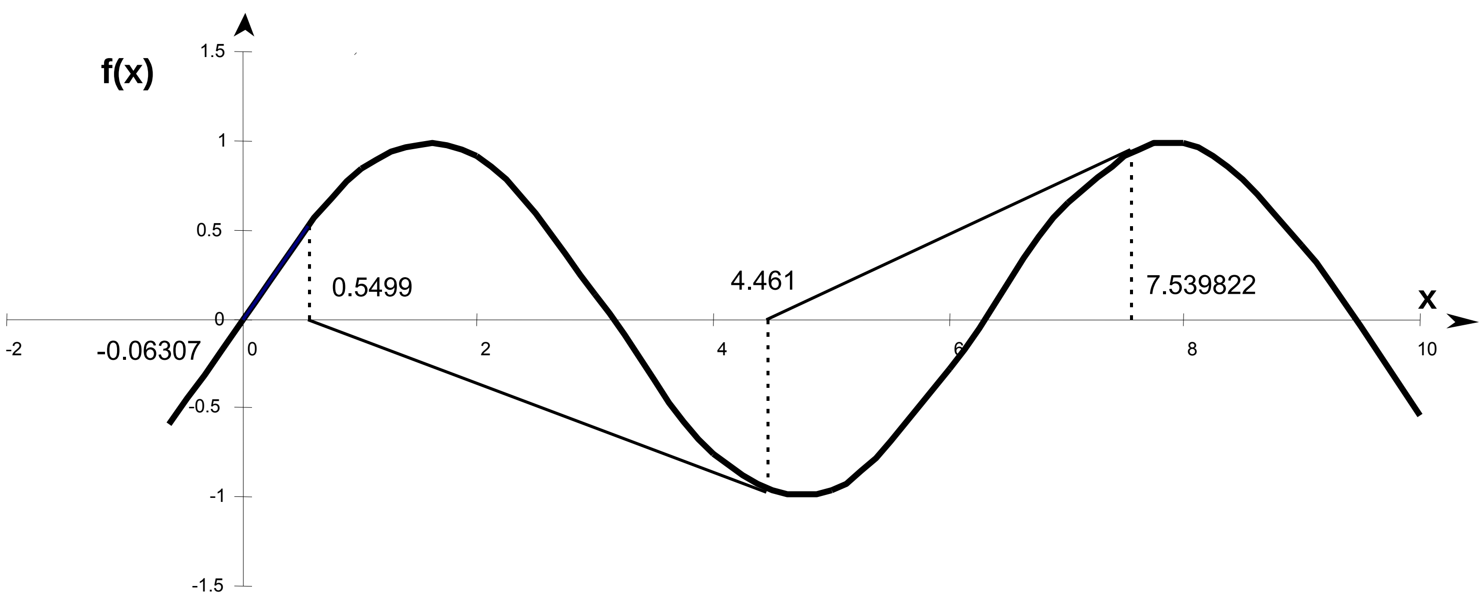


Figure 11 Root jumping from intended location of root for $f(x) = \sin x = 0$.

Script file: Newtraph.m

```
function root = newtraph(func,dfunc,xr,es,maxit)

%  newtraph(func,dfunc,xr,es,maxit);
%    uses Newton-Raphson method to find root of a function
%  input:
%    func = name of function
%    dfunc = name of derivative of function
%    xguess = initial guess
%    es = (optional) stopping criterion (%)
%    maxit = (optional) maximum allowable iterations
%  output:
%    root = real root
%  if necessary, assign default values
if nargin < 5, maxit = 50; end   % if maxit blank, set to 50
if nargin < 4, es = 0.001; end   % if es blank, set to 0.001

% Newton-Raphson
iter = 0;
while (1)
    xrold = xr;
    xr = xr - func(xr) / dfunc(xr);
    iter = iter + 1;
    if xr ~= 0, ea = abs((xr - xrold)/xr) * 100; end
    if ea <= es | iter >= maxit, break, end
end
root = xr;
```

Exercises

1. Use Newton-Raphson's Method to find a root of the equation correct to 2 decimal places.($\varepsilon = 0.01$)

$$x^3 - 2x - 5 = 0$$

$$f(x) = x^3 - 2x - 5$$

$$f'(x) = 3x^2 - 2$$

Result: 2.094551482

2. Use Newton-Raphson's Method to find a root of the equation correct to 2 decimal places.($\varepsilon = 0.01$)

$$x \sin x = -\cos x$$

$$f(x) = x \sin x + \cos x$$

$$f'(x) = x \cos x$$

Result: 2.798386046

Secant Method

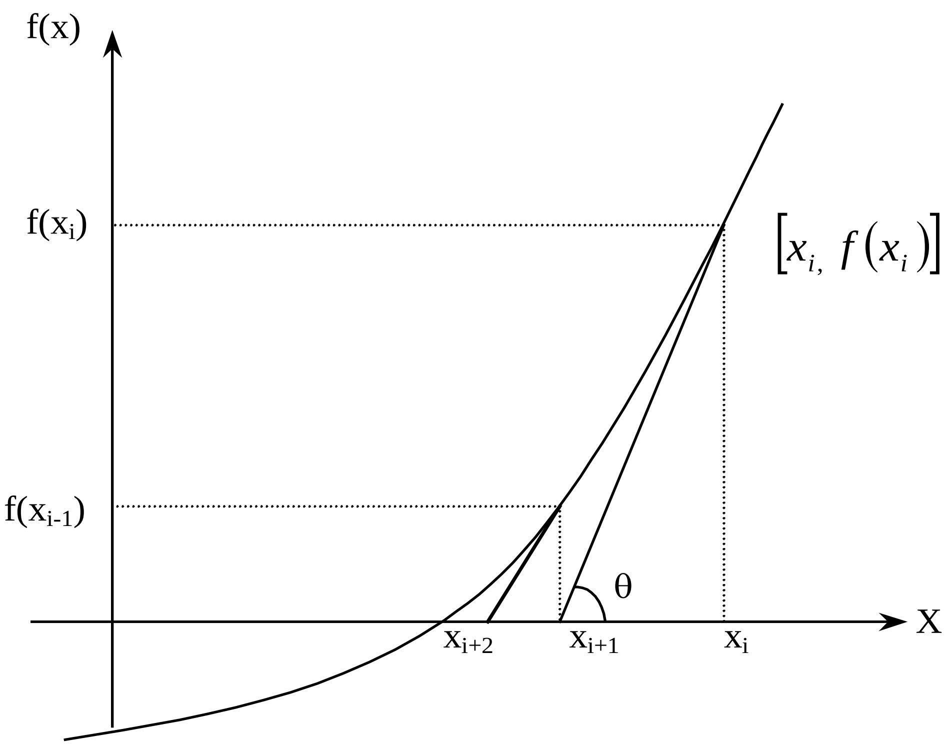


Figure 1 Geometrical illustration of the Newton-Raphson method.

Newton's Method

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

Approximate the derivative

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} \qquad (2)$$

Substituting Equation (2) into Equation (1) gives the Secant method

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

Derivation

The secant method can also be derived from geometry:

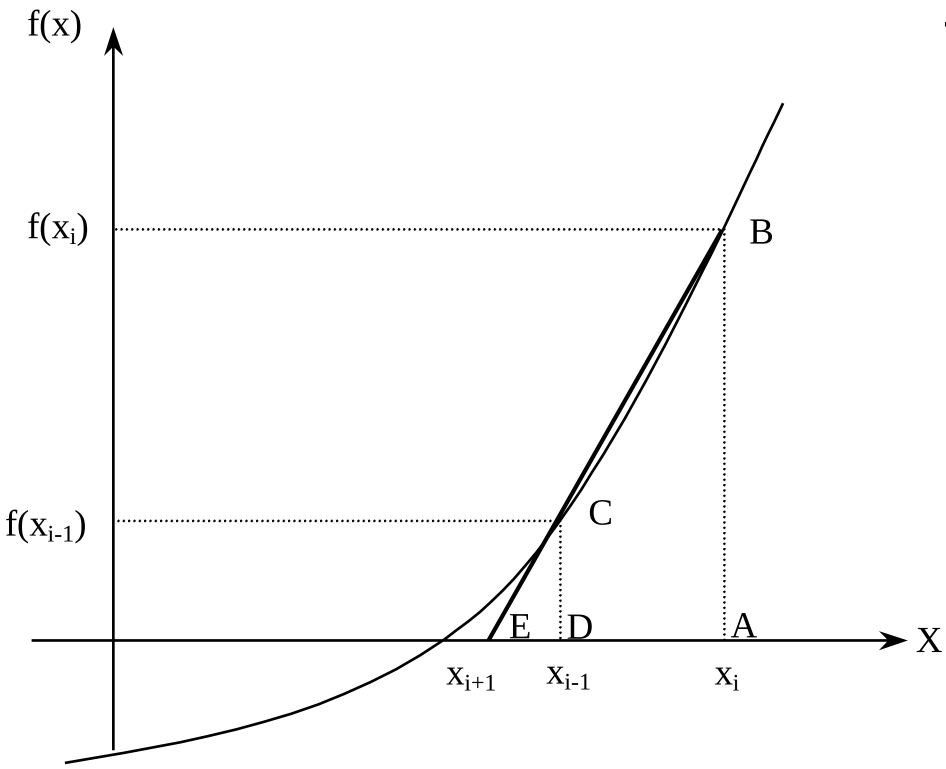


Figure 2 Geometrical representation of the Secant method.

The Geometric Similar Triangles

$$\frac{AB}{EA} = \frac{DC}{ED}$$

can be written as

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

On rearranging, the secant method is given as

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

Algorithm for Secant Method

Step 1 Calculate the next estimate of the root from two initial guesses

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

Find the absolute relative approximate error

$$\left|\in_a\right| = \left|\frac{x_{i+1} - x_i}{x_{i+1}}\right| \times 100$$

Step 2 Find if the absolute relative approximate error is greater than the pre-specified relative error tolerance.

If so, go back to step 1, else stop the algorithm.

Also check if the number of iterations has exceeded the maximum number of iterations.

Example 1

The equation that gives the depth *x* to which the ball is submerged under water is given by

$$f(x) = x^3 - 0.165x^2 + 3.993\times10^{-4}$$

Use the Secant method of finding roots of equations to find the depth *x* to which the ball is submerged under water.

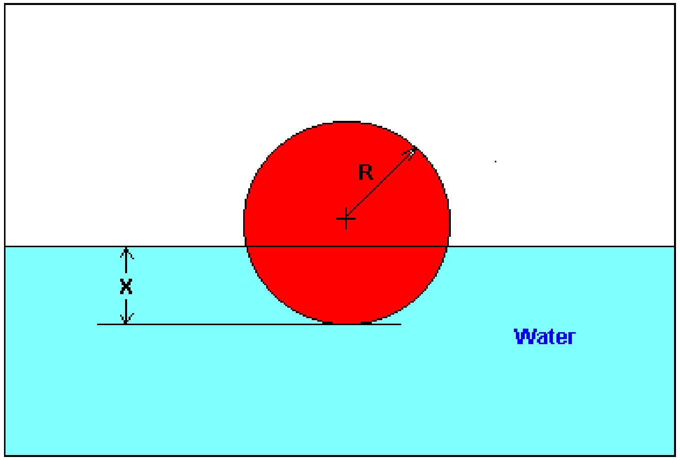


Figure 3 Floating Ball Problem.

- Conduct three iterations to estimate the root of the above equation.
- Find the absolute relative approximate error and the number of significant digits at least correct at the end of each iteration.

Solution

To aid in the understanding of how this method works to find the root of an equation, the graph of f(x) is shown, where

$$f(x) = x^3 - 0.165x^2 + 3.993\times10^{-4}$$

Let us assume the initial guesses of the root of $f(x) = 0$ as $x_0 = 0.05$.and $x_{-1} = 0.02$

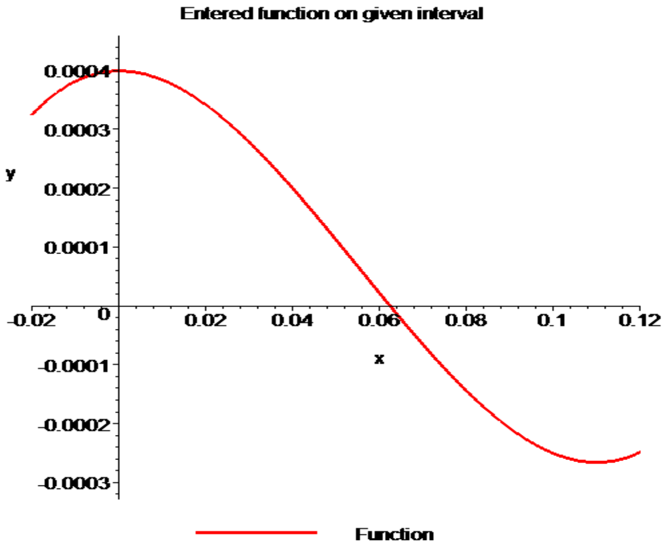


Figure 4 Graph of the function f(x).

Iteration 1

The estimate of the root is

$$x_1 = x_0 - \frac{f(x_0)(x_0 - x_{-1})}{f(x_0) - f(x_{-1})} = 0.05 - \frac{\left(0.05^3 - 0.165(0.05)^2 + 3.993\times10^{-4}\right)(0.05 - 0.02)}{\left(0.05^3 - 0.165(0.05)^2 + 3.993\times10^{-4}\right) - \left(0.02^3 - 0.165(0.02)^2 + 3.993\times10^{-4}\right)}$$

$$= 0.06461$$

The absolute relative approximate error $|\in_a|$ at the end of Iteration 1 is

$$|\in_a| = \left|\frac{x_1 - x_0}{x_1}\right| \times 100$$

$$= \left|\frac{0.06461 - 0.05}{0.06461}\right| \times 100$$

$$= 22.62\%$$

The number of significant digits at least correct is 0, for an absolute relative approximate error of 5% or less.

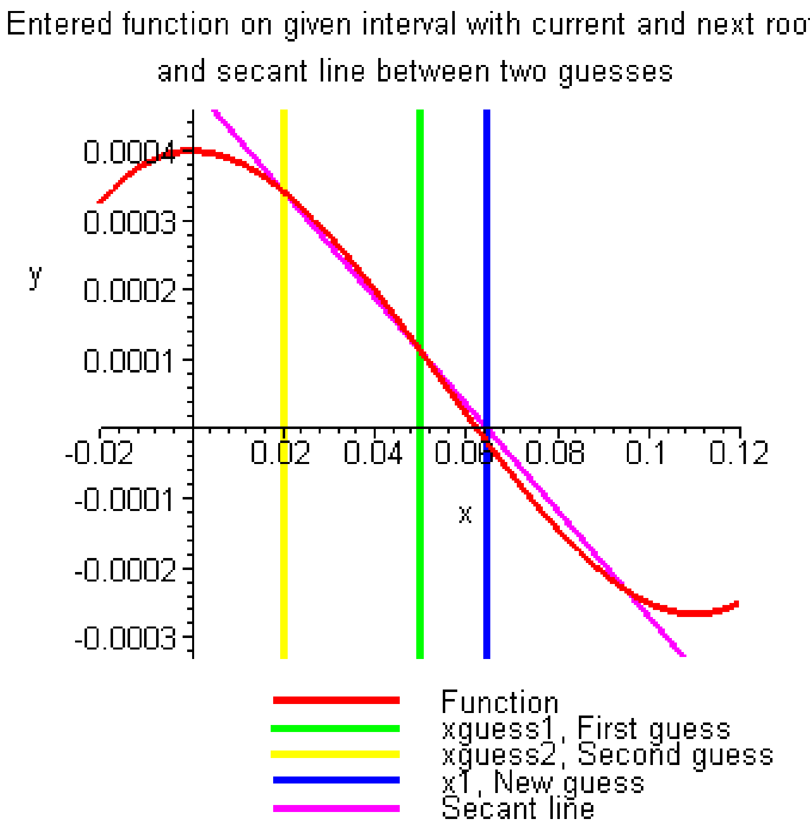


Figure 5 Graph of results of Iteration 1.

Iteration 2 The estimate of the root is

$$x_2 = x_1 - \frac{f(x_1)(x_1 - x_0)}{f(x_1) - f(x_0)}$$

$$= 0.06461 - \frac{\left(0.06461^3 - 0.165(0.06461)^2 + 3.993\times10^{-4}\right)(0.06461 - 0.05)}{\left(0.06461^3 - 0.165(0.06461)^2 + 3.993\times10^{-4}\right) - \left(0.05^3 - 0.165(0.05)^2 + 3.993\times10^{-4}\right)}$$

$$= 0.06241$$

The absolute relative approximate error $|\in_a|$ at the end of Iteration 2 is

$$|\in_a| = \left|\frac{x_2 - x_1}{x_2}\right| \times 100$$

$$= \left|\frac{0.06241 - 0.06461}{0.06241}\right| \times 100$$

$$= 3.525\%$$

The number of significant digits at least correct is 1 for an absolute relative approximate error of 5% or less.

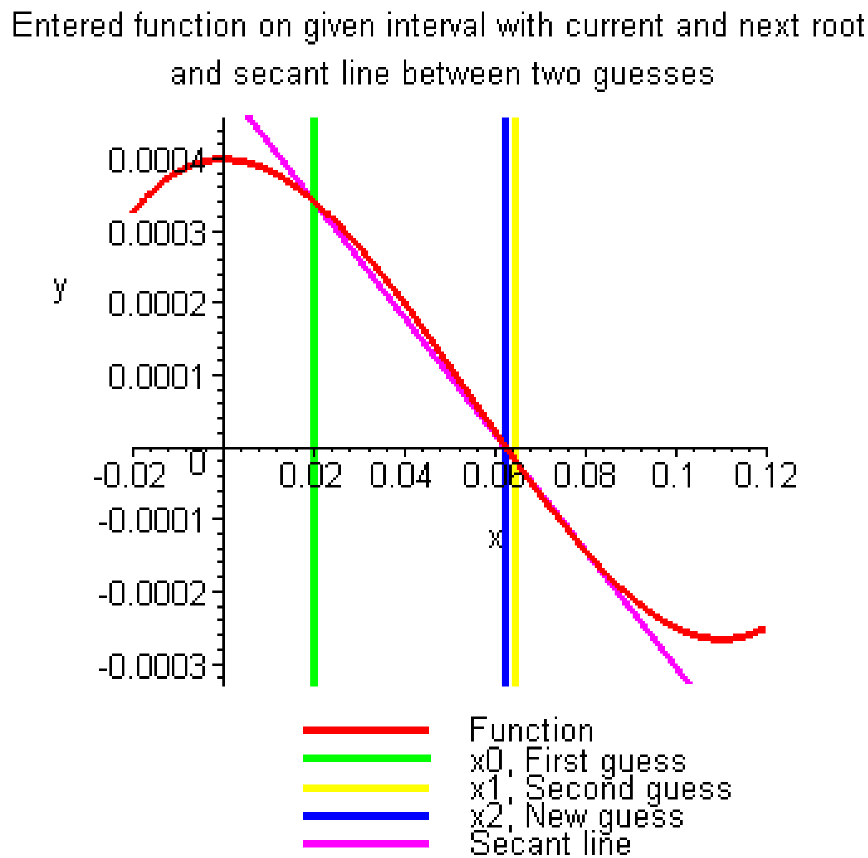


Figure 6 Graph of results of Iteration 2.

Iteration 3 The estimate of the root is

$$x_3 = x_2 - \frac{f(x_2)(x_2 - x_1)}{f(x_2) - f(x_1)}$$

$$= 0.06241 - \frac{\left(0.06241^3 - 0.165(0.06241)^2 + 3.993\times10^{-4}\right)(0.06241 - 0.06461)}{\left(0.06241^3 - 0.165(0.06241)^2 + 3.993\times10^{-4}\right) - \left(0.05^3 - 0.165(0.06461)^2 + 3.993\times10^{-4}\right)}$$

$$= 0.06238$$

The absolute relative approximate error $|\in_a|$ at the end of Iteration 3 is

$$|\in_a| = \left|\frac{x_3 - x_2}{x_3}\right| \times 100$$

$$= \left|\frac{0.06238 - 0.06241}{0.06238}\right| \times 100$$

$$= 0.0595\%$$

The number of significant digits at least correct is 2 for an absolute relative approximate error of 5% or less.

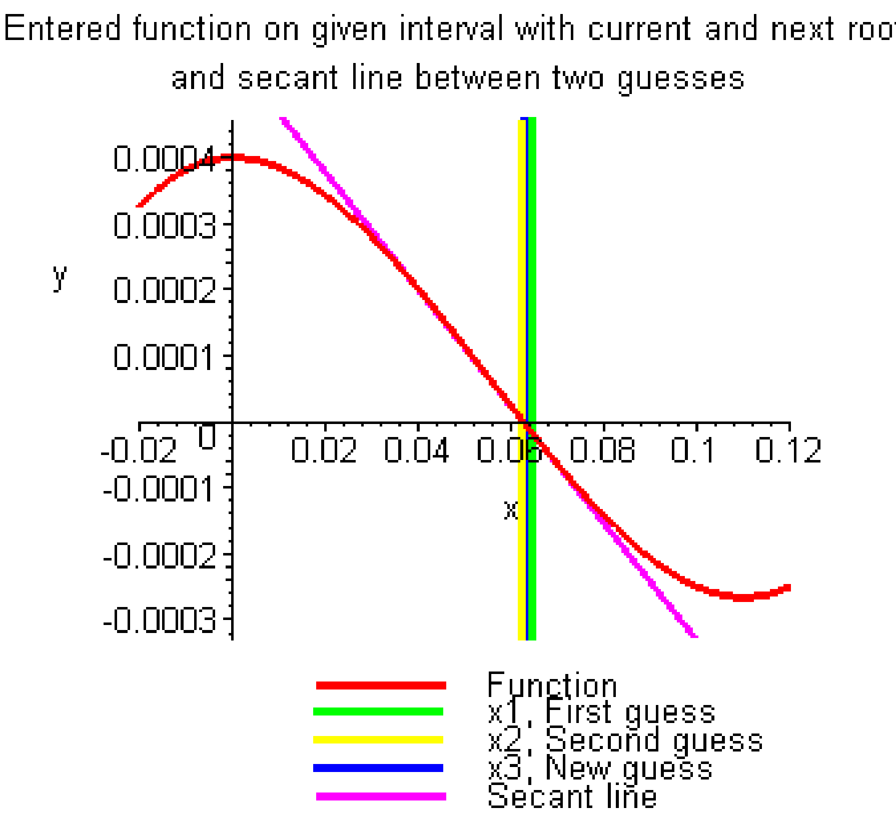


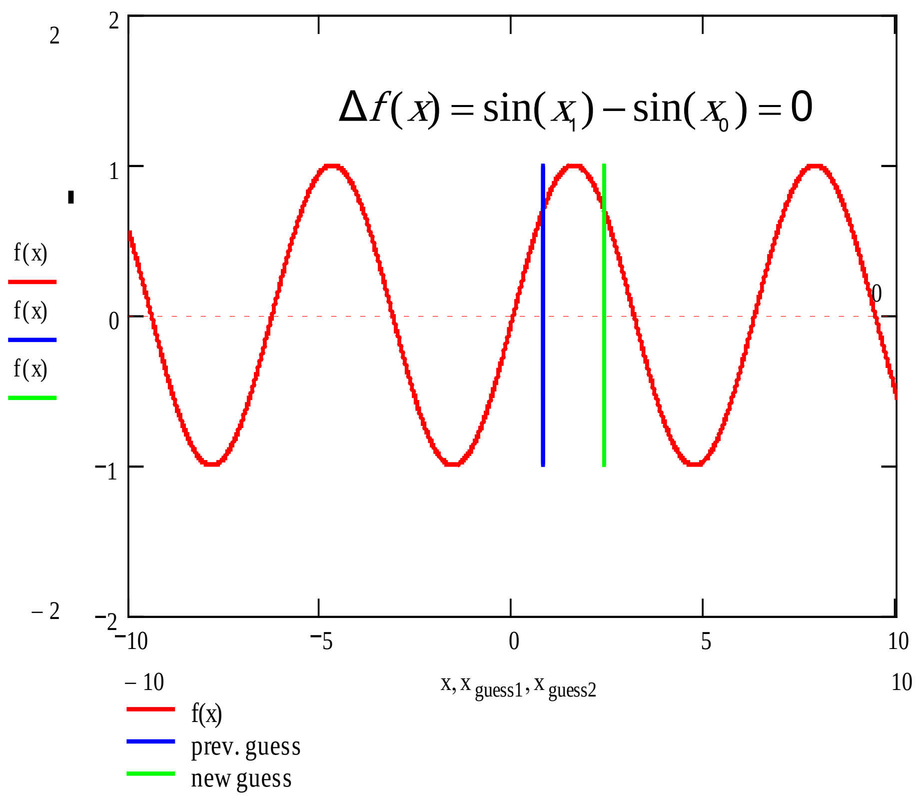
Figure 7 Graph of results of Iteration 3.

Advantages and Drawbacks

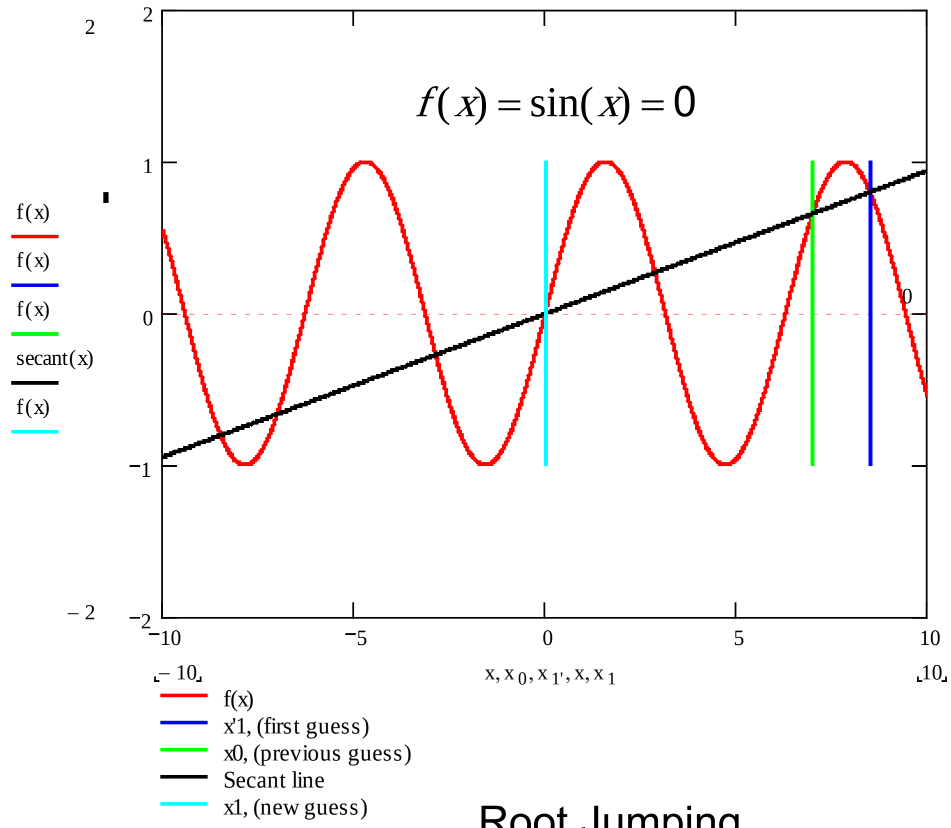
Advantages

- Converges fast, if it converges
- Requires two guesses that do not need to bracket the root

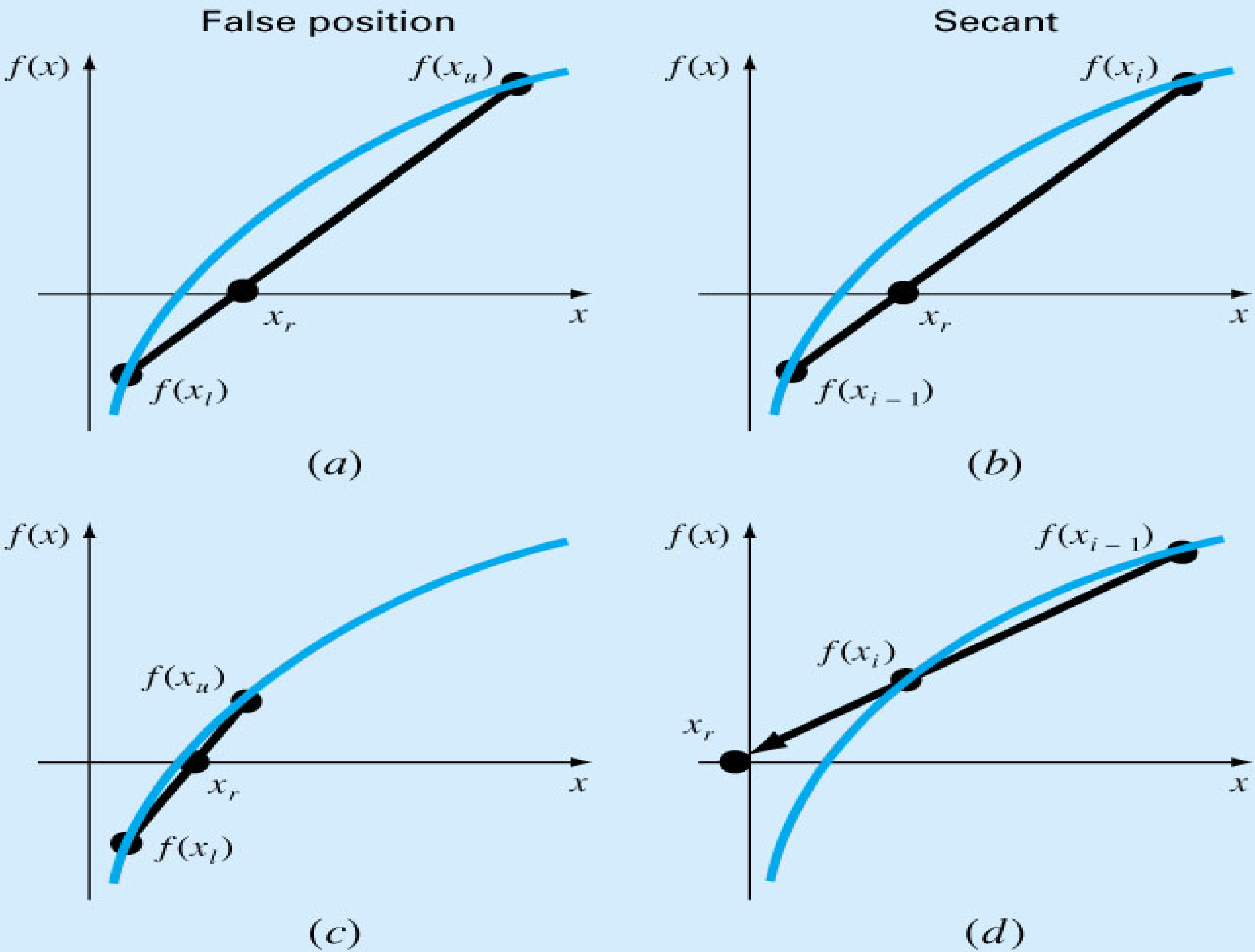
Drawbacks



Division by zero



Root Jumping



False position
Secant
$f(x)$
$f(x_u)$
x_r
x
$f(x_l)$
$f(x_i)$
$f(x_{i-1})$
(a)
(b)
(c)
(d)

Comparison between F-P & Secant Methods

False-point method

- Starting two points.
- Similar formula

  $x_m = x_u - f(x_u)(x_u - x_l)/(f(x_u) - f(x_l))$

- Next iteration:

  points replacement: if

  $f(x_m)*f(x_l) < 0$, then

  $x_u = x_m$ else $x_l = x_m$.
- Require bracketing.
- Always converge

Secant method

- Starting two points.
- Similar formula

  $x_3 = x_2 - f(x_2)(x_2 - x_1)/(f(x_2) - f(x_1))$

- Next iteration:

  points replacement: always

  $x_1 = x_2$ & $x_2 = x_3$.
- no requirement of bracketing.
- Faster convergence
- May not converge

Secant method

$$f(x) = x^3 - 3x + 1 = 0$$

» [x1 f1]=secant('my_func',0,1,1.e-15,100);

secant method has converged

step	x	f
1.0000	0	1.0000
2.0000	1.0000	-1.0000
3.0000	0.5000	-0.3750
4.0000	0.2000	0.4080
5.0000	0.3563	-0.0237
6.0000	0.3477	-0.0011
7.0000	0.3473	0.0000
8.0000	0.3473	0.0000
9.0000	0.3473	0.0000
10.0000	0.3473	0.0000

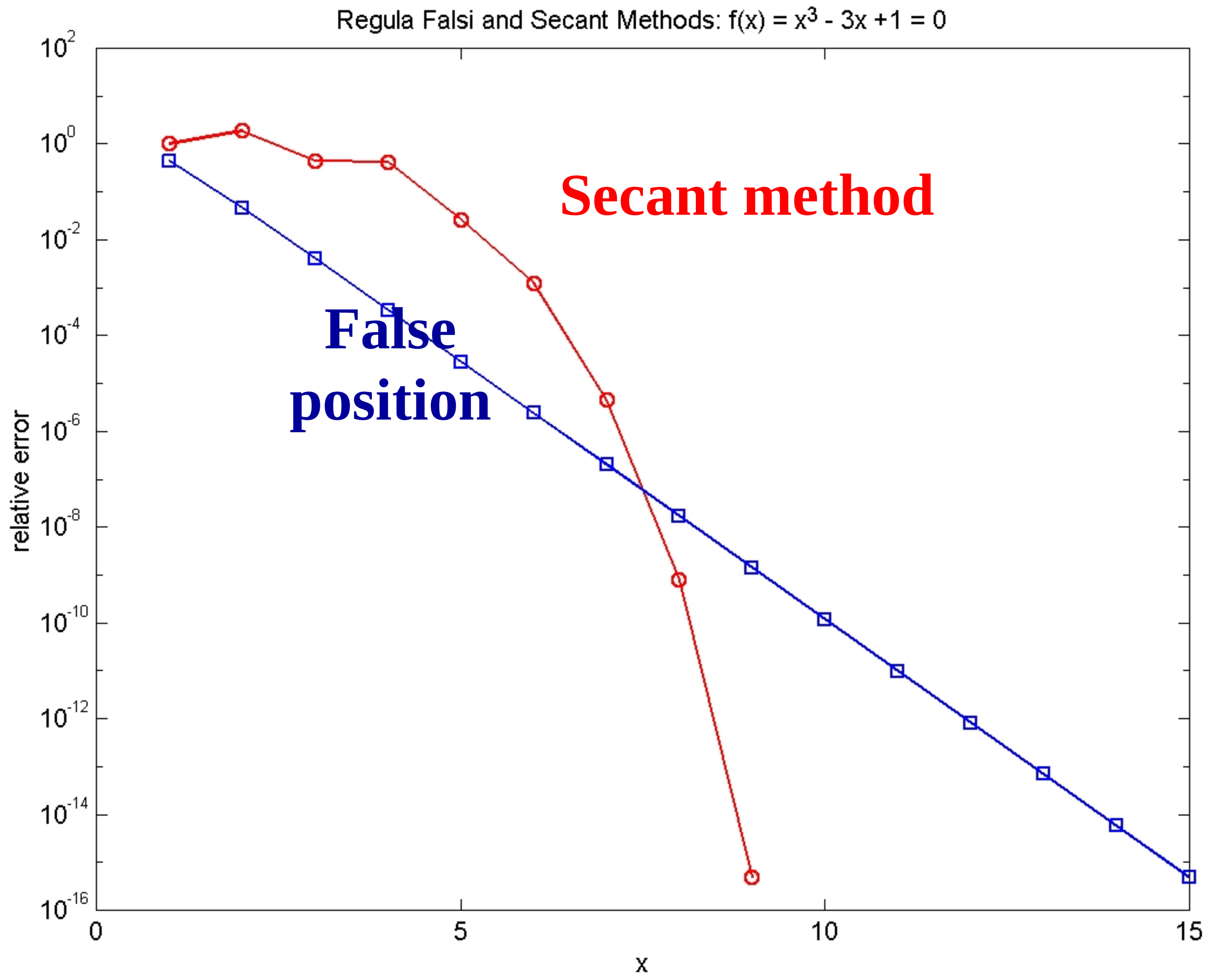
False position method

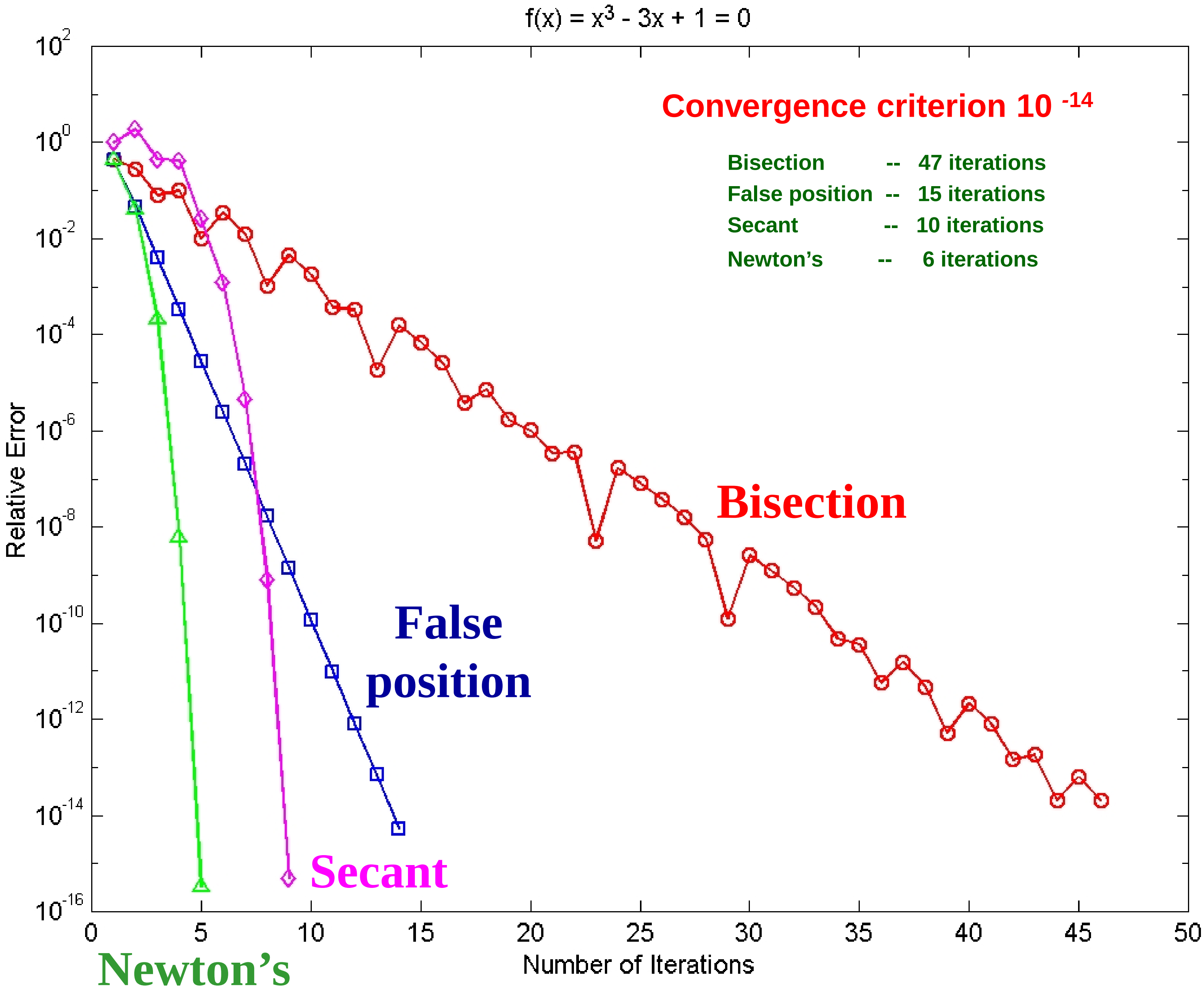
» [x2 f2]=false_position('my_func',0,1,1.e-15,100);

false_position method has converged

step	xl	xu	x	f
1.0000	0	1.0000	0.5000	-0.3750
2.0000	0	0.5000	0.3636	-0.0428
3.0000	0	0.3636	0.3487	-0.0037
4.0000	0	0.3487	0.3474	-0.0003
5.0000	0	0.3474	0.3473	0.0000
6.0000	0	0.3473	0.3473	0.0000
7.0000	0	0.3473	0.3473	0.0000
8.0000	0	0.3473	0.3473	0.0000
9.0000	0	0.3473	0.3473	0.0000
10.0000	0	0.3473	0.3473	0.0000
11.0000	0	0.3473	0.3473	0.0000
12.0000	0	0.3473	0.3473	0.0000
13.0000	0	0.3473	0.3473	0.0000
14.0000	0	0.3473	0.3473	0.0000
15.0000	0	0.3473	0.3473	0.0000
16.0000	0	0.3473	0.3473	0.0000

Secant method may converge even faster and it doesn't need to bracket the root





f(x) = x3 - 3x + 1 = 0
Convergence criterion 10 -14
Bisection -- 47 iterations
False position -- 15 iterations
Secant -- 10 iterations
Newton's -- 6 iterations
Bisection
False position
Secant
Newton's
Relative Error
Number of Iterations
10^2
10^0
10^-2
10^-4
10^-6
10^-8
10^-10
10^-12
10^-14
10^-16
0
5
10
15
20
25
30
35
40
45
50

Summary of Methods

- **Bisection Method**
  - Condition: Continuous function $f(x)$ *in* $[a, b]$ *satisfying* $f(a)\, f(b) < 0$.
  - Convergence: Slow but sure. Linear.
- **False position method**
  - Condition: Continuous function $f(x)$ *in* $[a, b]$ *satisfying* $f(a)\, f(b) < 0$.
  - Convergence: Slow (linear).
- **Fixed-point Method**
  - Condition: Contraction of $g(x)$.
  - Convergence: Varying with the nature of $g(x)$.
- **Newton Method**
  - Condition: Existence of nonzero $f'(x)$
  - Convergence: Fast (quadratic).
- **Secant Method**
  - Condition: Existence of nonzero $f(x_{n+1}) - f(x_n)$
  - Convergence: Fast (quadratic).

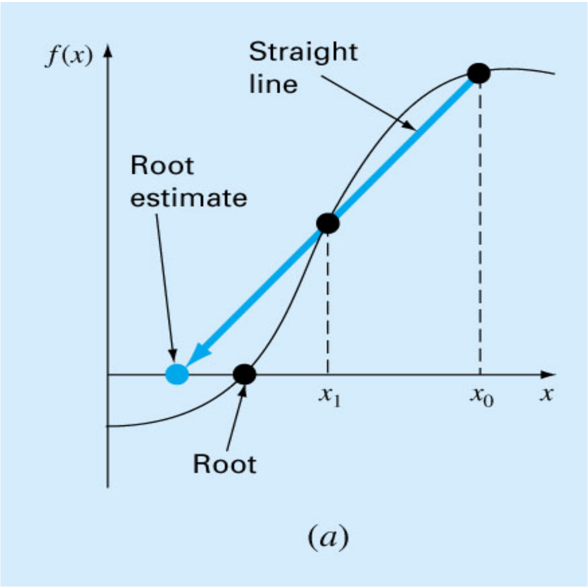
Muller's Method

This is a method to find the roots of equations polynomials in the general way:

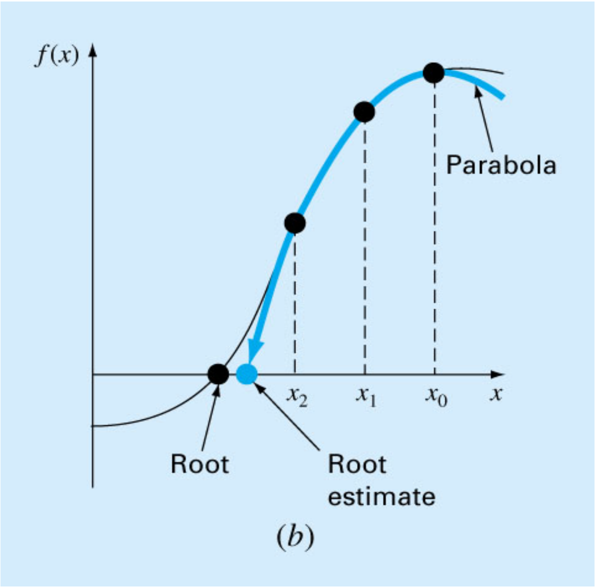
$$f_n(x) = a_0 + a_1 x + a_2 x^2 + \ldots\ldots + a_n x^n$$

Where *n* is the order of the polynomial and they are constant coefficients.

The method consists on obtaining the coefficients of the three points, to substitute them in the quadratic formula to obtain the point where the parabola intercepts the axis x.



Secant Method



Muller's Method

Solution by Muller's Method

The method consists of deriving the coefficients of parabola that goes through the three points:

1. Write the equation in a convenient form:

$$f_2(x) = a(x - x_2)^2 + b(x - x_2) + c$$

2. The parabola should intersect the three points $[x_o, f(x_o)]$, $[x_1, f(x_1)]$, $[x_2, f(x_2)]$.

   The coefficients of the polynomial can be estimated by substituting three points to give

$$x = x_0: \quad f(x_o) = a(x_o - x_2)^2 + b(x_o - x_2) + c$$

$$x = x_1: \quad f(x_1) = a(x_1 - x_2)^2 + b(x_1 - x_2) + c$$

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

3. Three equations can be solved for three unknowns *a*, *b*, *c*. Since two of the terms in the 3rd equation are zero, it can be immediately solved for $c = f(x_2)$.

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

If $\quad h_o = x_1 - x_o \qquad h_1 = x_2 - x_1$

$$\delta_o = \frac{f(x_1) - f(x_o)}{x_1 - x_o} \qquad \delta_1 = \frac{f(x_2) - f(x_1)}{x_2 - x_1}$$

$$\left.\begin{aligned} (h_o + h_1)b - (h_o + h_1)^2 a &= h_o\delta_o + h_1\delta_1 \\ h_1 b - h_1^2 a &= h_1\delta_1 \end{aligned}\right\} \text{Solved for } a \text{ and } b$$

$$a = \frac{\delta_1 - \delta_o}{h_1 + h_o} \qquad b = ah_1 + \delta_1 \qquad c = f(x_2)$$

Roots can be found by applying an alternative form of quadratic formula:

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

The error can be calculated as

$$\varepsilon_a = \left|\frac{x_3 - x_2}{x_3}\right| 100\%$$

± term yields two roots. This will result in a largest denominator, and will give root estimate that is closest to x_2.

Once x_3 is determined, the process is repeated using the following guidelines:

1. If only real roots are being located, choose the two original points that are nearest the new root estimate, x_3.
2. If both real and complex roots are estimated, employ a sequential approach just like in secant method, x_1, x_2, and x_3 to replace x_o, x_1, and x_2.

Example : Use Muller's method to find roots of $f(x) = x^3 - 13x - 12$

Initial guesses of x_0, x_1, and x_2 of 4.5, 5.5 and 5.0 respectively.

(*Roots are -3, -1 and 4*)

Solution :

$f(x_0) = f(4.5) = 20.625$ and $f(x_1) = f(5.5) = 82.875$

$f(x_2) = f(5) = 48$

$h_0 = 5.5 - 4.5 = 1$, $h_1 = 5 - 5.5 = -0.5$

$$\delta_0 = \frac{82.875 - 20.625}{5.5 - 4.5} = 62.25 \quad , \quad \delta_1 = \frac{48 - 82.875}{5 - 5.5} = 69.75$$

$$a = \frac{69.75 - 62.25}{-0.5 + 1} = 15 \ , \ b = 15(-0.5) + 69.75 = 62.25 \ , \ c = 48$$

(Choose sign similar to the sign of *b*)

$$\sqrt{b^2 - 4ac} = \sqrt{62.25^2 - 4 \times 15 \times 48} = 31.54461$$

$$x_3 = 5 + \frac{-2 \times 48}{62.25 + 31.54461} = 3.976487$$

$$\varepsilon_a = \left| \frac{-1.023513}{x_3} \right| \times 100\% = 25.74\%$$

The second iteration will have x_0= 5.5 , x_1= 5 and x_2= 3.976487

Iteration	x_r	Error %
0	5	-
1	3.976487	25.7
2	4.001	0.614
3	4.000	0.026
4	4.000	0.000012

Muller’s Method Program

C:\MATLAB6p5p1\work\mullerquick.m

```
function mullerquick(f,x1,x2,x3)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Given three initial guesses x1, x2, and x3 compute       %
% y1 = f(x1), y2 = f(x2), y3 = f(x3) and                   %
% w=-2*y3/(s+sign(s)*sqrt(s^2-4*d1*y3))+x3                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
k = 1;
while abs(x2-x3) > eps*abs(x3)
   y1=f(x1);
   y2=f(x2);
   y3=f(x3);
   c1=(y1-y2)/(x1-x2);
   c2=(y2-y3)/(x2-x3);
   d1=(c1-c2)/(x1-x3);
   s=d1*(x3-x2)+c2;
   w=-2*y3/(s+sign(s)*sqrt(s^2-4*d1*y3))+x3;
   if sign(imag(w)) == +1
       s = '+';
   else
       s = '-';
   end
   fprintf(['x(' num2str(k) ') = %16.14f %1s %16.14fi\n'],real(w), s, abs(imag(w)))
   x1=x2;
   x2=x3;
   x3=w;
   k = k +1;
end
```