NUMERICAL METHODS

Lecture 1

Dr. P V Ramana

How does a CPU compute the following functions for a specific x value?

$$cos(x) \qquad sin(x) \qquad e^x \qquad log(x) \qquad \text{etc.}$$

- Non-elementary functions such as ***trigonometric***, ***exponential***, and others are expressed in an approximate fashion using **Taylor series** when their values, derivatives, and integrals are computed.

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

Taylor Series (n^{th} *order* approximation):

$$f(x_{i+1}) \cong f(x_i) + \frac{f^{'}(x_i)}{1!}(x_{i+1} - x_i) + \frac{f^{''}(x_i)}{2!}(x_{i+1} - x_i)^2 + \mathrm{K} + \frac{f^{(n)}(x_i)}{n!}(x_{i+1} - x_i)^n + R_n$$

The Reminder term, R_n, accounts for all terms from (n+1) to infinity.

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

Define the *step size* as **$h=(x_{i+1}- x_i)$**, the ***series*** becomes:

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

cos(x) ***sin(x)*** ***e^x*** ***log(x)*** etc.

Any smooth function can be approximated as a polynomial.

Take $\boldsymbol{x = x_{i+1}}$ Then $\boldsymbol{f(x) \approx f(x_i)}$ ***zero order*** approximation

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

first order approximation

Second order approximation:

$$f(x) \cong f(x_i) + \frac{f^{'}(x_i)}{1!}(x - x_i) + \frac{f^{''}(x_i)}{2!}(x - x_i)^2$$

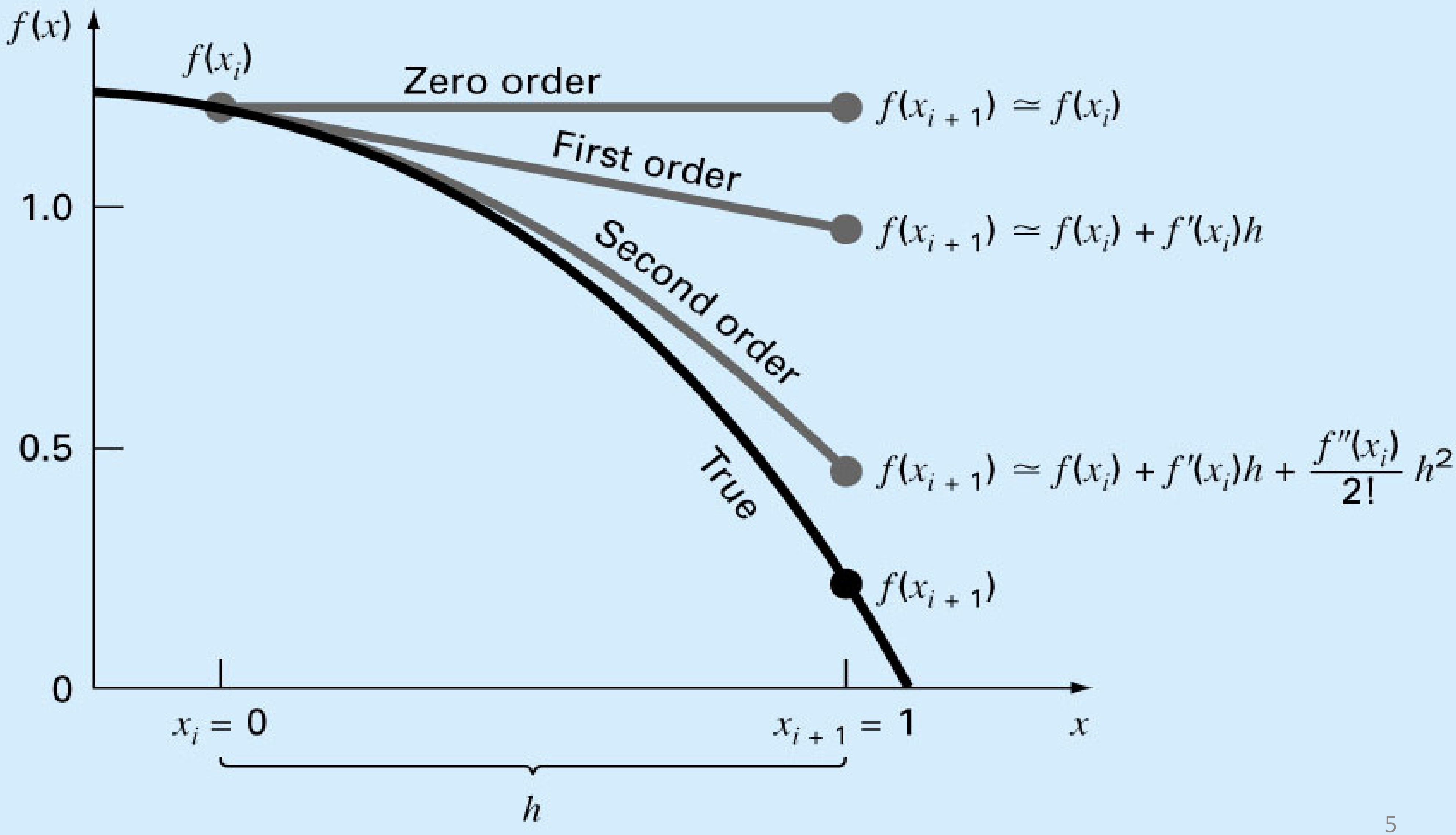
n^{th} order approximation:

$$f(x) \cong f(x_i) + \frac{f^{'}(x_i)}{1!}(x - x_i) + \frac{f^{''}(x_i)}{2!}(x - x_i)^2 + \mathrm{K} + \frac{f^{(n)}(x_i)}{n!}(x - x_i)^n + R_n$$

- Each additional term will contribute some improvement to the approximation. Only if an infinite number of terms are added will the series yield an exact result.
- **In most cases, only a few terms will result in an approximation that is close enough to the true value for practical purposes**

Example

Approximate the function $f(x) = 1.2 - 0.25x - 0.5x^2 - 0.15x^3 - 0.1x^4$ from $x_i = 0$ with $h = 1$ and **predict** $f(x)$ at $x_{i+1} = 1$.



Roots of Equations: Bracketing Methods

- Easy

$$ax^2 + bx + c = 0 \quad \Rightarrow \quad x = \frac{-b \,\mu \sqrt{b^2 - 4ac}}{2a}$$

- But, not easy

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

- How about these?

$$\sin x + x = 0 \quad \Rightarrow x = ?$$

$$\cos(10x) + \sin(3x) = 0 \quad \Rightarrow x = ?$$

Roots of Equations

Our first real numerical method –

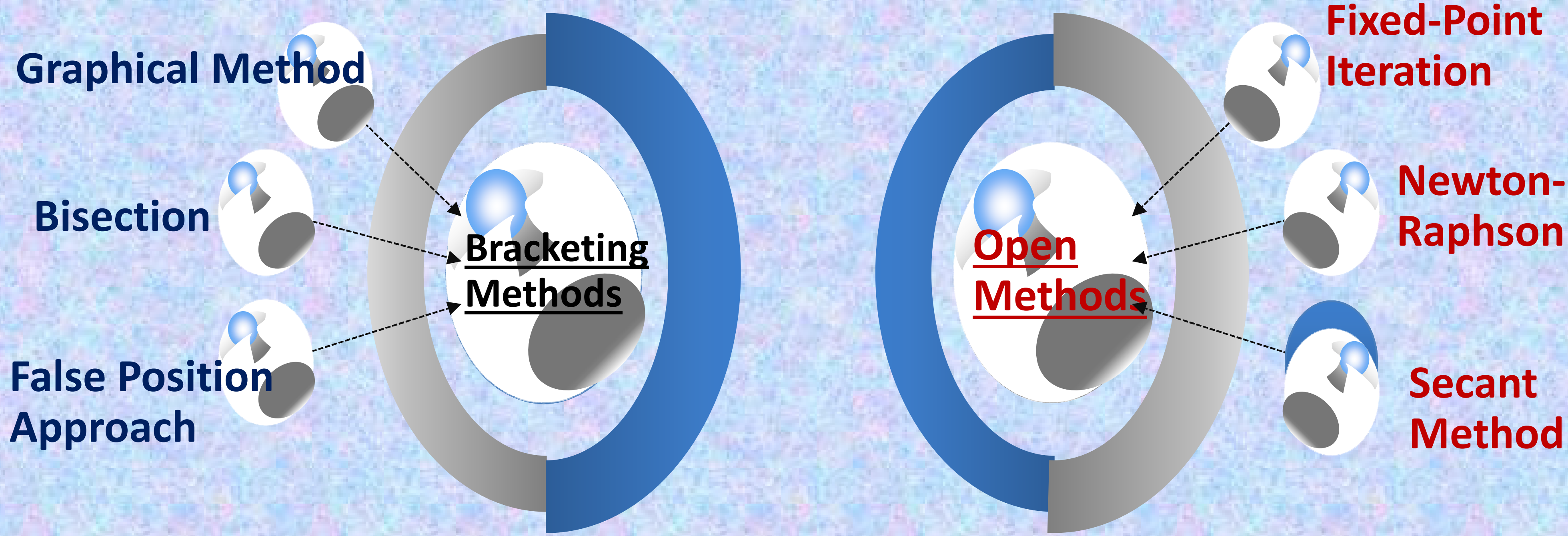
Root finding

Finding the value *x* where a function

y = f(x) = 0

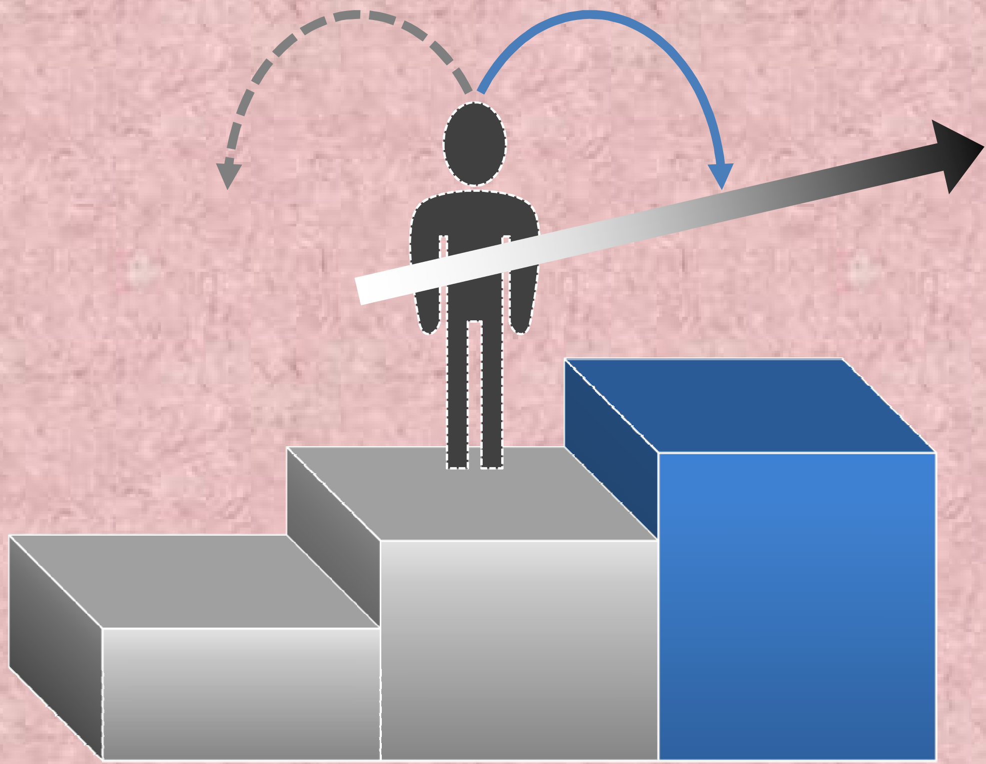
Encounter above process again and again

Two Fundamental Approaches



Root Finding Problems

Many problems in Science and Engineering are expressed as:



Given a continuous function $f(x)$, find the value r such that $f(r) = 0$

These problems are called root finding problems.

A number *r* that satisfies an equation is called a root of the equation.

The equation :

$x^4 - 3x^3 - 7x^2 + 15x = -18$

has four roots :

$-2, 3, 3, and\ -1$.

i.e., $x^4 - 3x^3 - 7x^2 + 15x + 18$

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

Roots of Equations

The equation

has two simple

roots (-1 and -2)

and a repeated root (3)

with multiplicity $= 2$.

Zeros of a Function

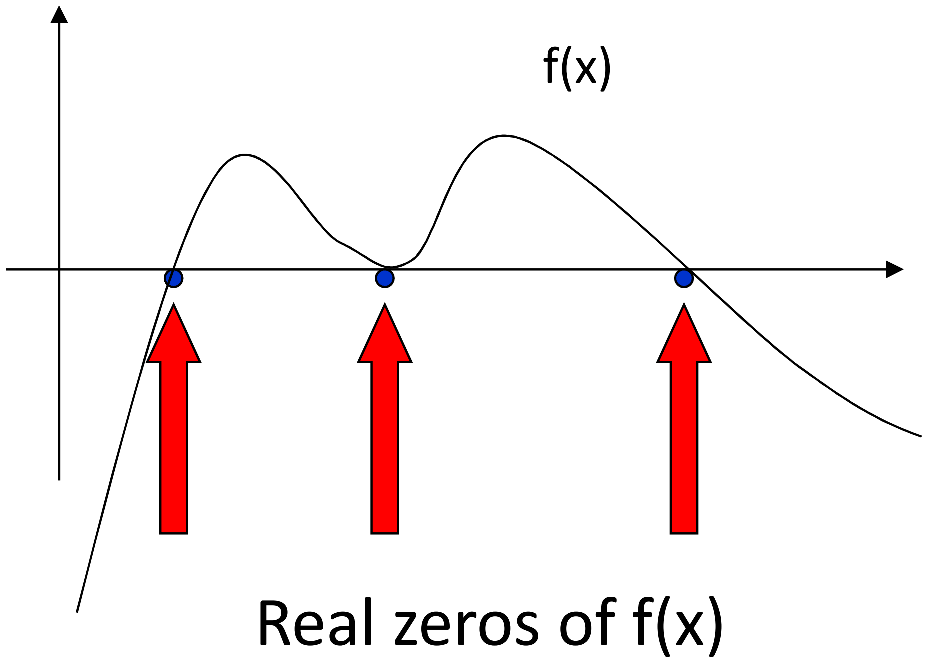
Let *f(x)* be a real-valued function of a real variable. Any number *r* for which *f(r)=0* is called a zero of the function.

Examples: The equation : $x^2 - 5x + 6 = 0$

2 and 3 are zeros of the function f(x) = (x-2)(x-3).

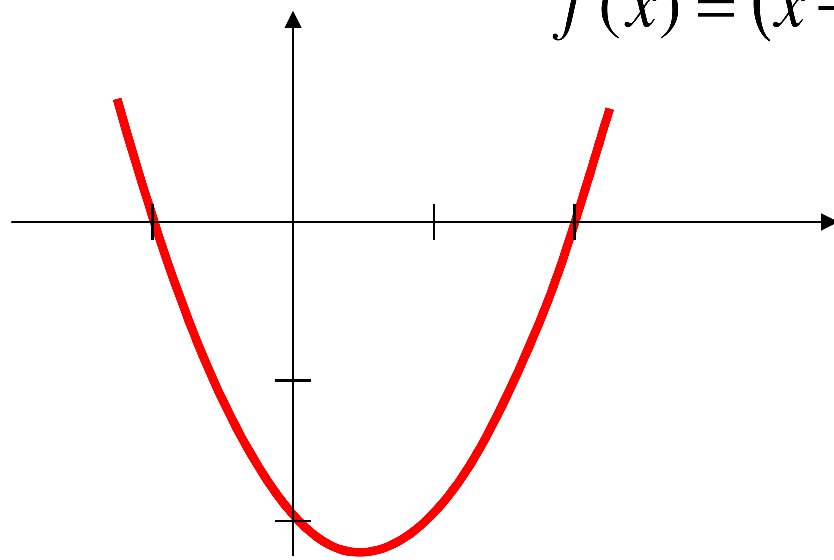
Graphical Interpretation of Zeros

- The real zeros of a function ***f(x)*** are the values of ***x*** at which the graph of the function crosses (or touches) the x-axis.



Simple Zeros

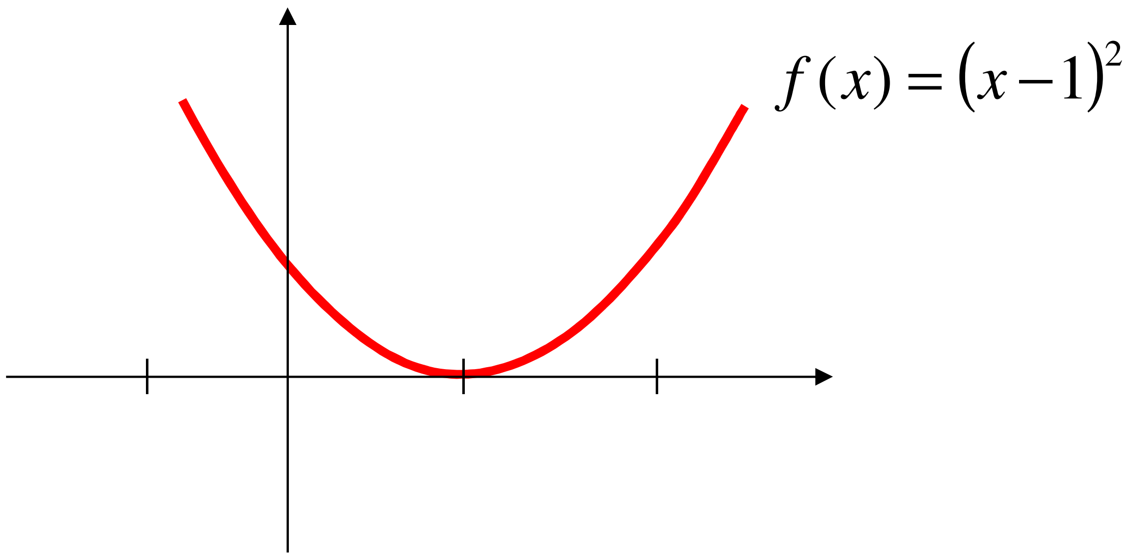
$$f(x) = (x+1)(x-2)$$



$$f(x) = (x+1)(x-2) = x^2 - x - 2$$

has two simple zeros (one at $x = 2$ and one at $x = -1$)

Multiple Zeros



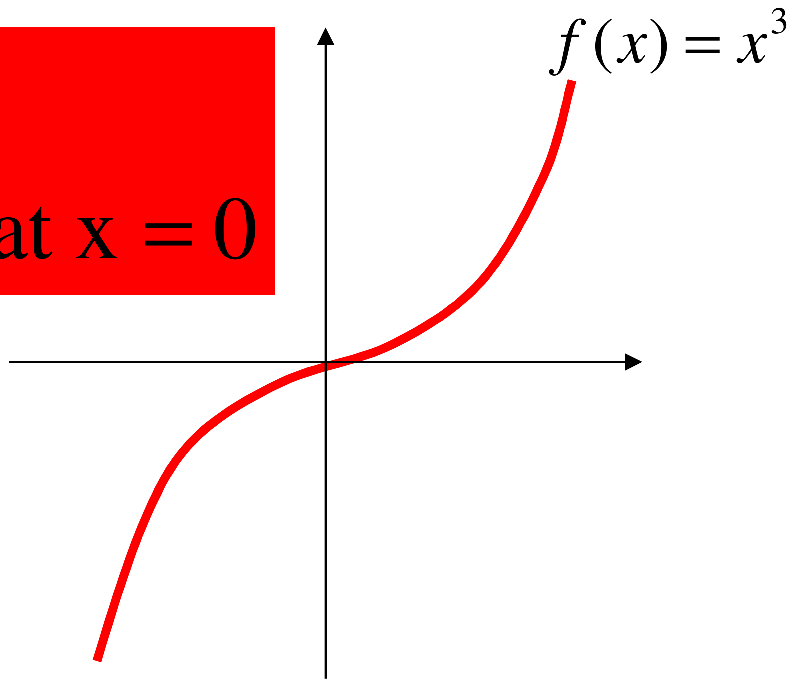
$$f(x) = (x-1)^2 = x^2 - 2x + 1$$

has double zeros (zero with muliplicity = 2) at x = 1

Multiple Zeros

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

has a zero with muliplicity = 2 at x = 0



$$f(x) = x^3$$

has a zero with muliplicity = 3 at x = 0

Roots of Equations & Zeros of Function

Given the equation :

$$x^4 - 3x^3 - 7x^2 + 15x = -18$$

Move all terms to one side of the equation :

$$x^4 - 3x^3 - 7x^2 + 15x + 18 = 0$$

Define $f(x)$ as :

$$f(x) = x^4 - 3x^3 - 7x^2 + 15x + 18$$

The zeros of $f(x)$ are the same as the roots of the equation $f(x) = 0$

(Which are -2, 3, 3, and -1)

Solution Methods

Several ways to solve nonlinear equations are possible:

Analytical Solutions

- Possible for simple equations only

Graphical Solutions

- Useful for providing initial guesses for other methods

Numerical Solutions for special

- Open methods
- Bracketing methods

Analytical Methods

Analytical Solutions are available for simple equations only.

$$\text{Analytical solution of}: \; a\,x^2 + b\,x + c = 0$$

$$roots = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$\text{No analytical solution is available for}: \quad x - e^{-x} = 0$$

Graphical Methods

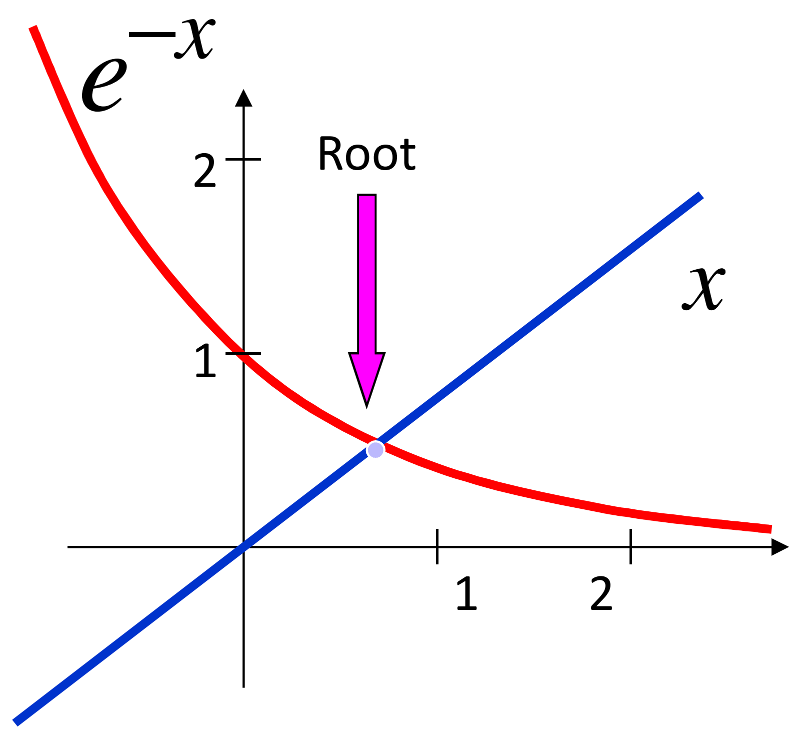
- Graphical methods are useful to provide an initial guess to be used by other methods.

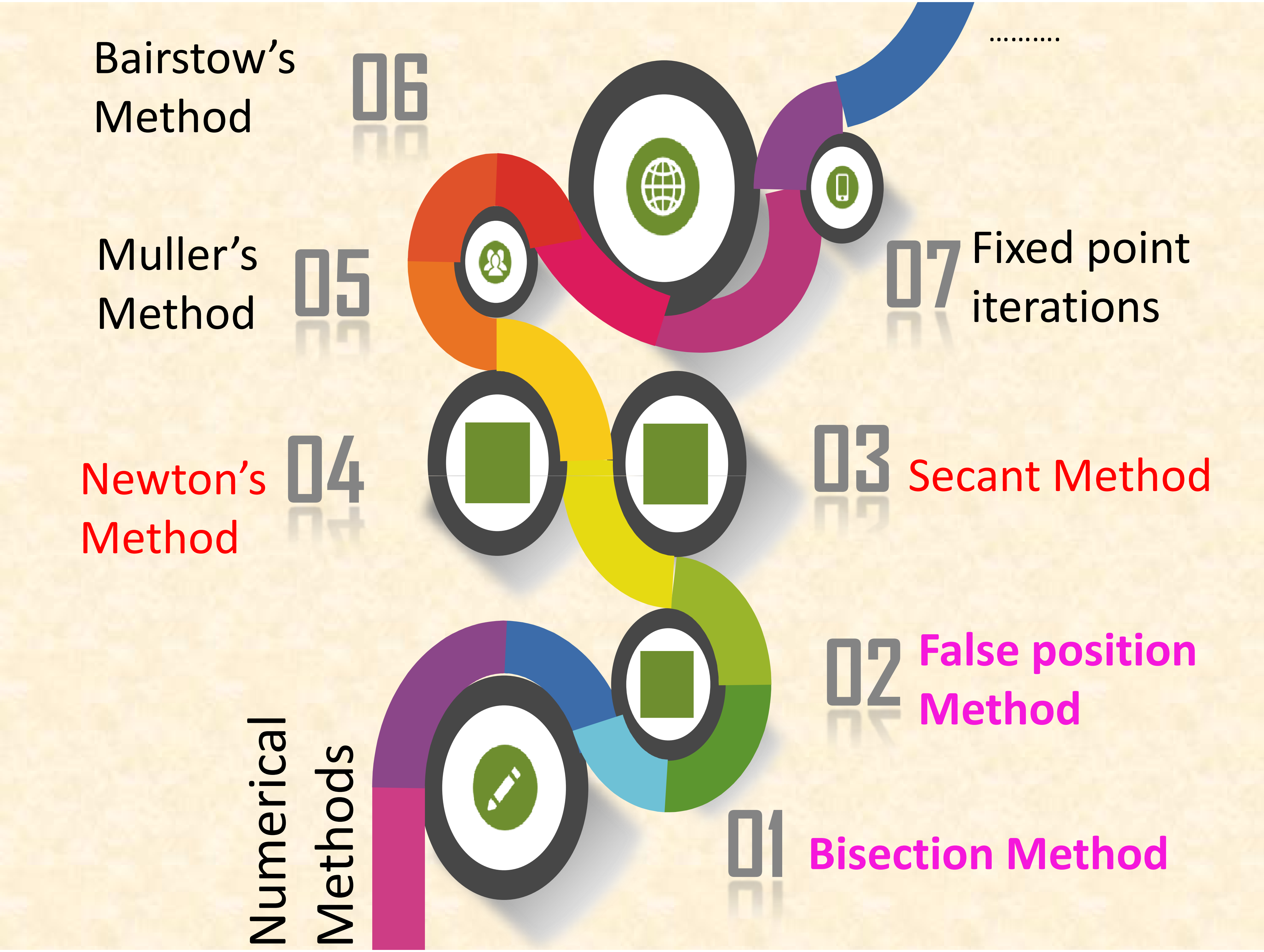
$$\mathit{Solve}$$

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

$$\mathit{The}\quad \mathit{root} \in [0,1]$$

$$\mathit{root} \approx 0.6$$




Numerical Methods
01
Bisection Method
02
False position Method
03
Secant Method
04
Newton's Method
05
Muller's Method
06
Bairstow's Method
07
Fixed point iterations
....

Bracketing Methods

In bracketing methods, the method starts with an interval that contains the root and a procedure is used to obtain a smaller interval containing the root.

Examples of bracketing methods:

- Bisection method
- False position method

Open Methods

- In the open methods, the method starts with one or more initial guess points. In each iteration, a new guess of the root is obtained.
- Open methods are usually more efficient than bracketing methods.
- They may not converge to a root.

Convergence Notation

A sequence $x_1, x_2, ..., x_n, ...$ is said to **converge** to x if to every $\varepsilon > 0$ there exists N such that :

$$|x_n - x| < \varepsilon \quad \forall n > N$$

Convergence Notation

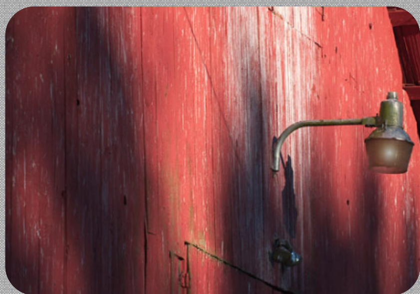
Let $x_1, x_2, \ldots,$ converge to x.

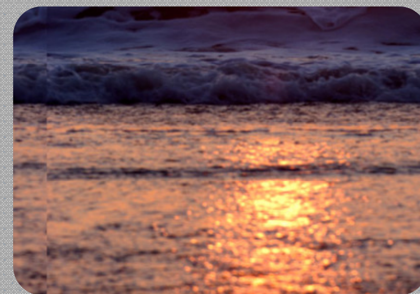
Linear Convergence: $\dfrac{|x_{n+1} - x|}{|x_n - x|} \leq C$

Quadratic Convergence: $\dfrac{|x_{n+1} - x|}{|x_n - x|^2} \leq C$

Convergence of order P: $\dfrac{|x_{n+1} - x|}{|x_n - x|^p} \leq C$

Speed of Convergence





One can compare different methods in terms of their convergence rate.

Quadratic convergence is faster than linear convergence.

A method with convergence order ***q*** converges faster than a method with convergence order ***p*** if ***q>p***.

Methods of convergence order ***p>1*** are said to have super linear convergenc

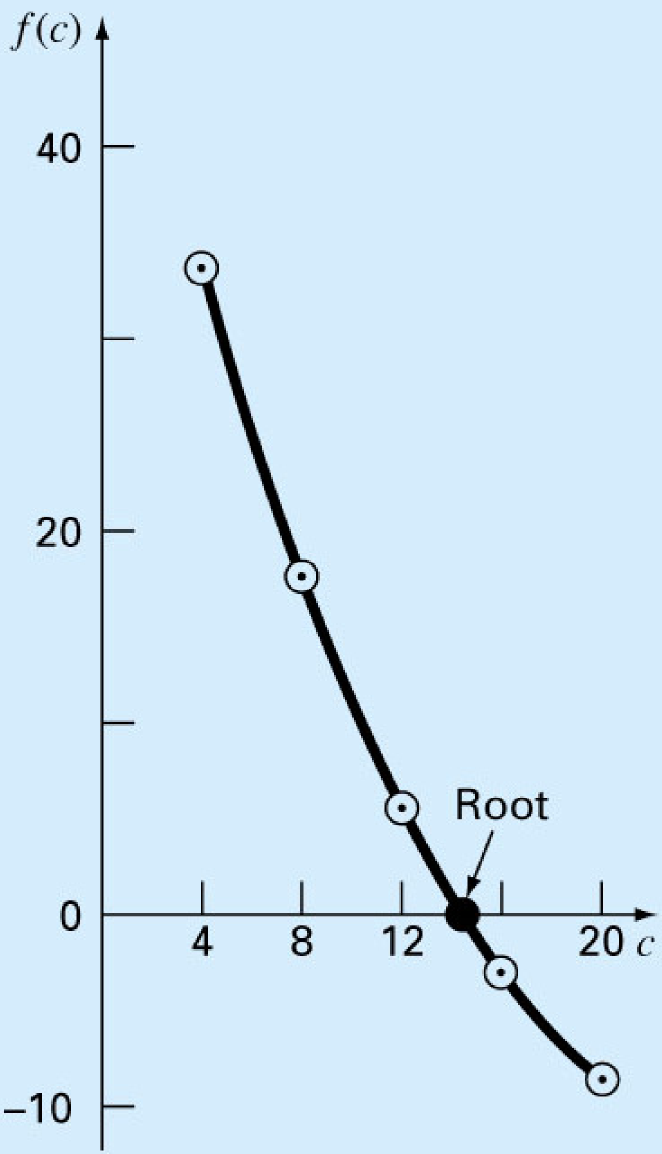
Graphical Methods

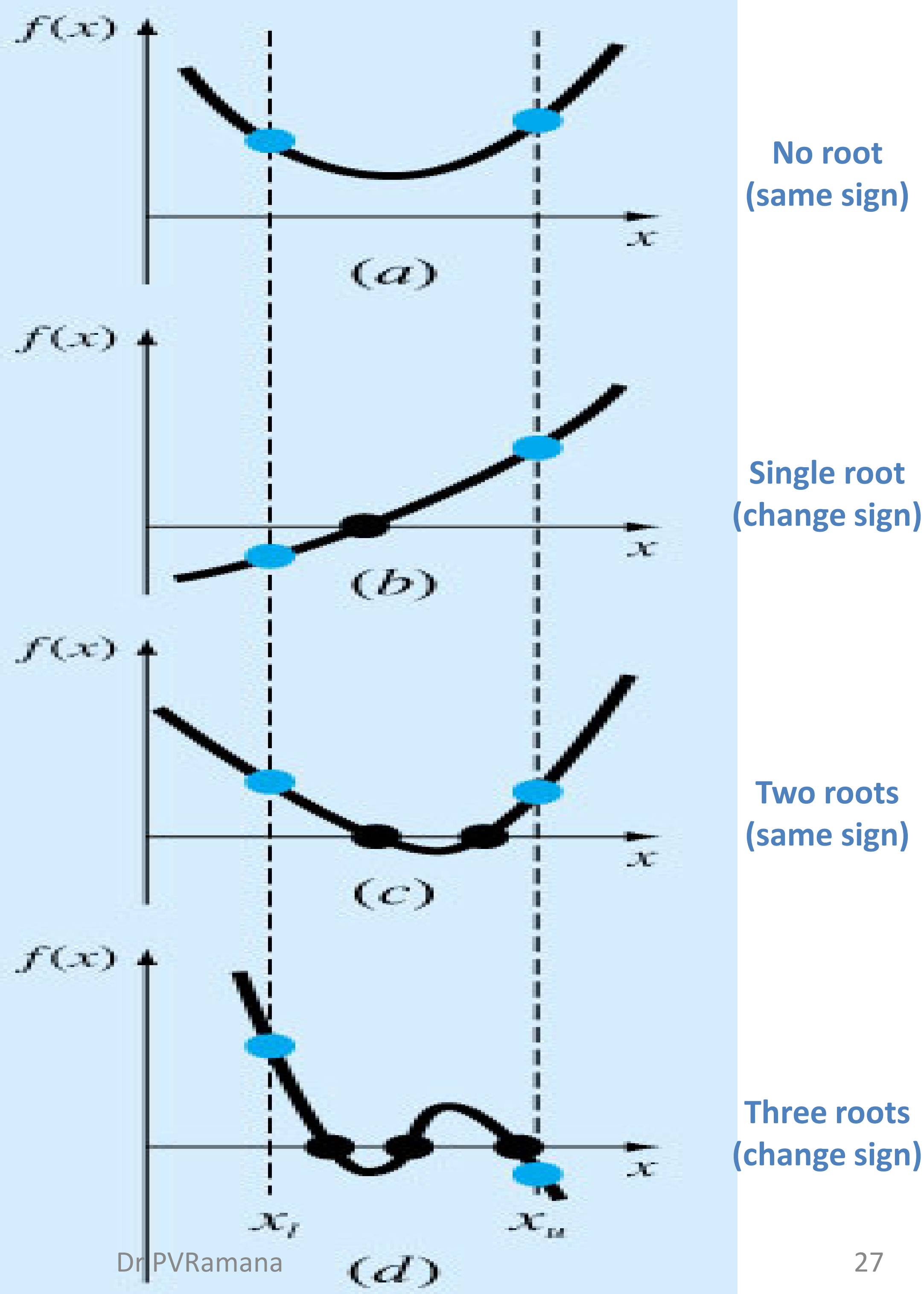
- **Progressive enlarge & by Inspection**

Bracketing Methods

- **Bisection**
- **False-Position**

Graphical methods

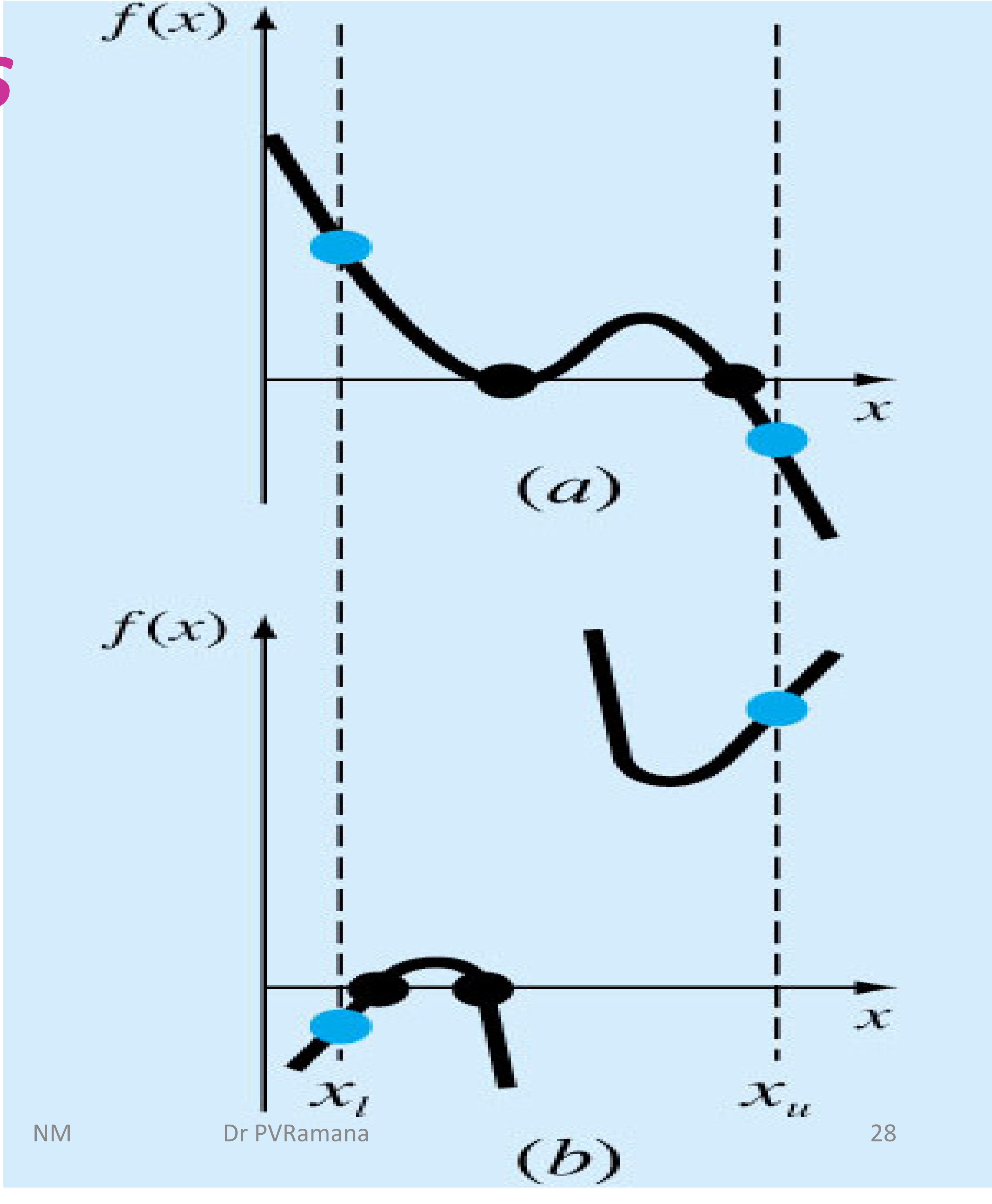




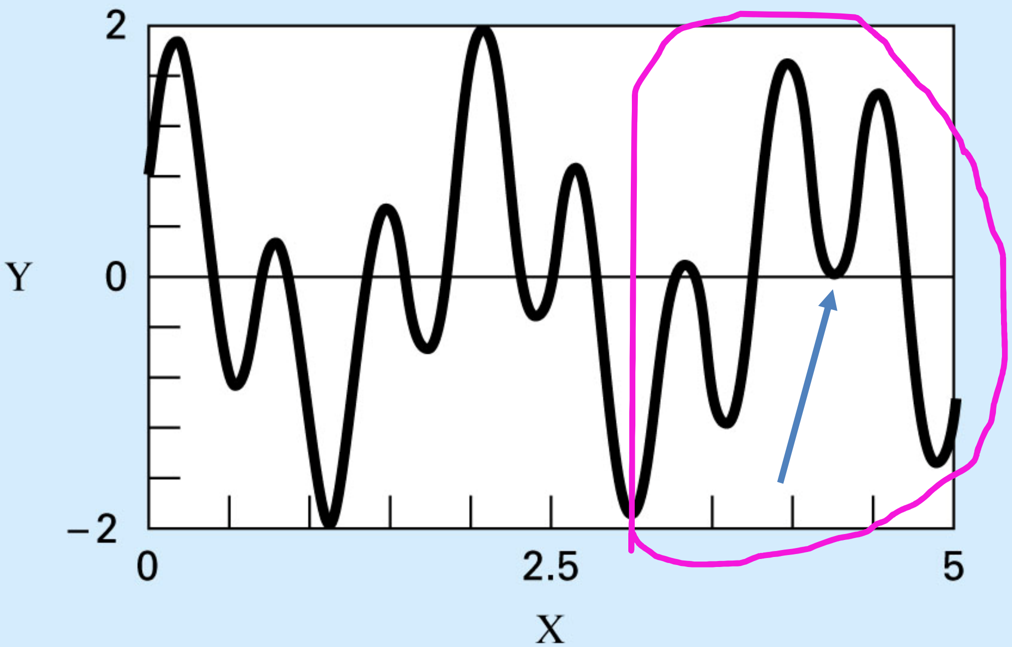
Special Cases

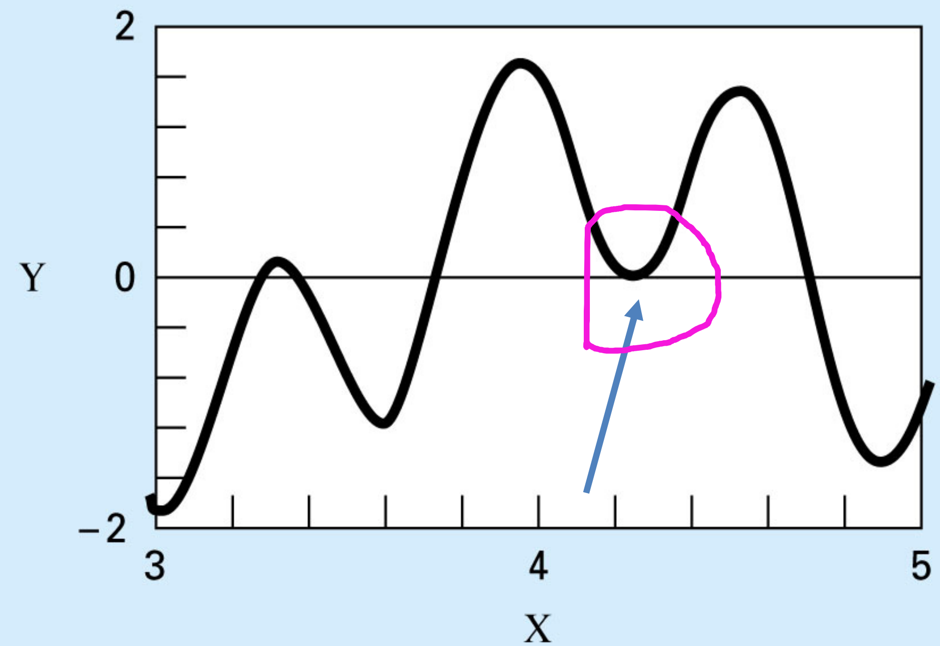
Multiple Roots

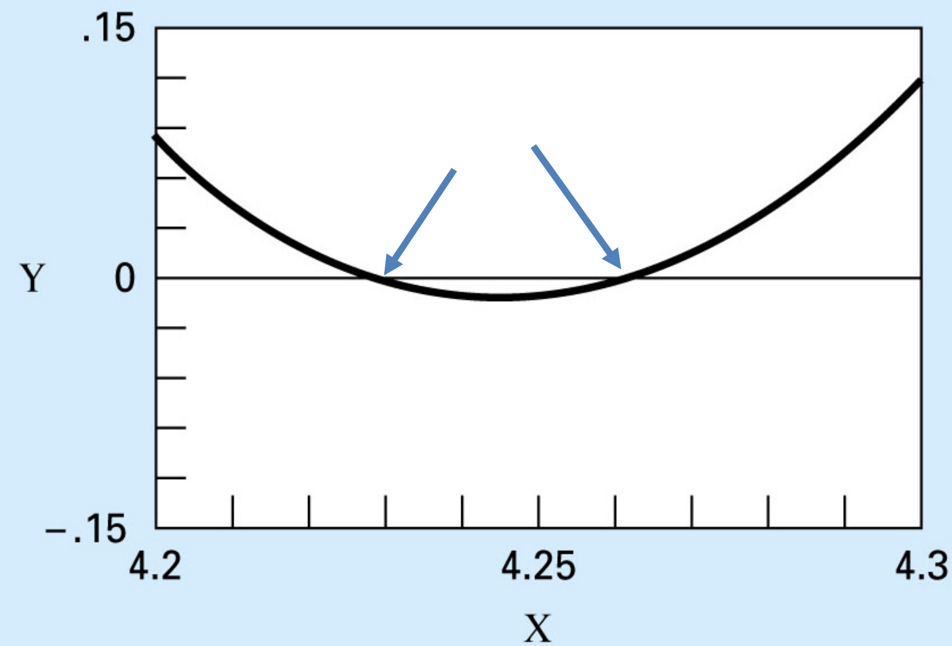
Discontinuity



Graphical Method - Progressive Enlargement







Two distinct roots

Graphical Method

Graphical method is useful for getting an idea of what’s going on in a problem, but depends on eyeball.

Use bracketing methods to improve the accuracy

Bisection and false-position methods

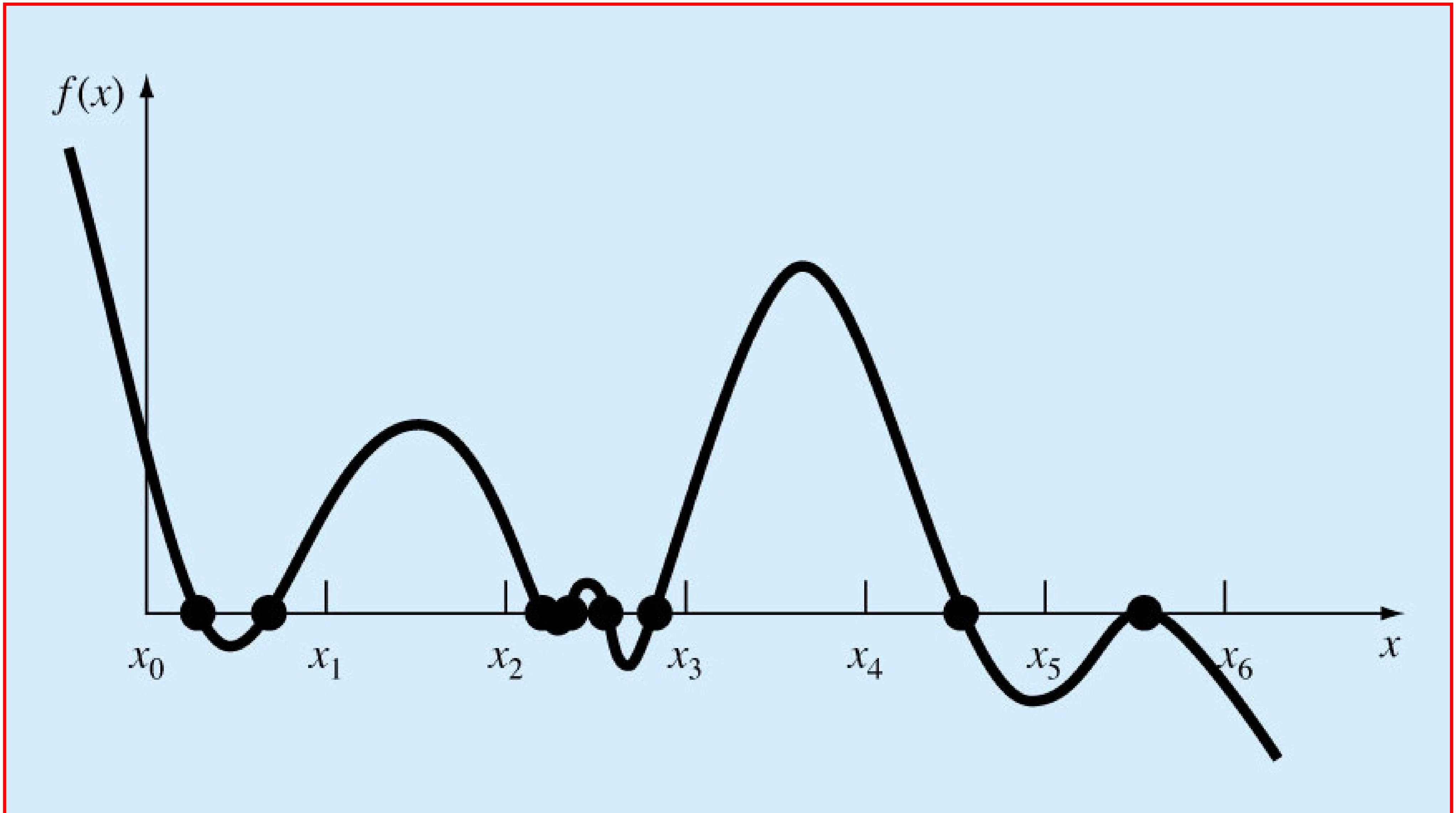
Both bisection and false-position methods require the root to be bracketed by the endpoints.

Bracketing Methods

How to find the endpoints?

* plotting the function
* incremental search
* trial and error

Incremental Search



Incremental Search

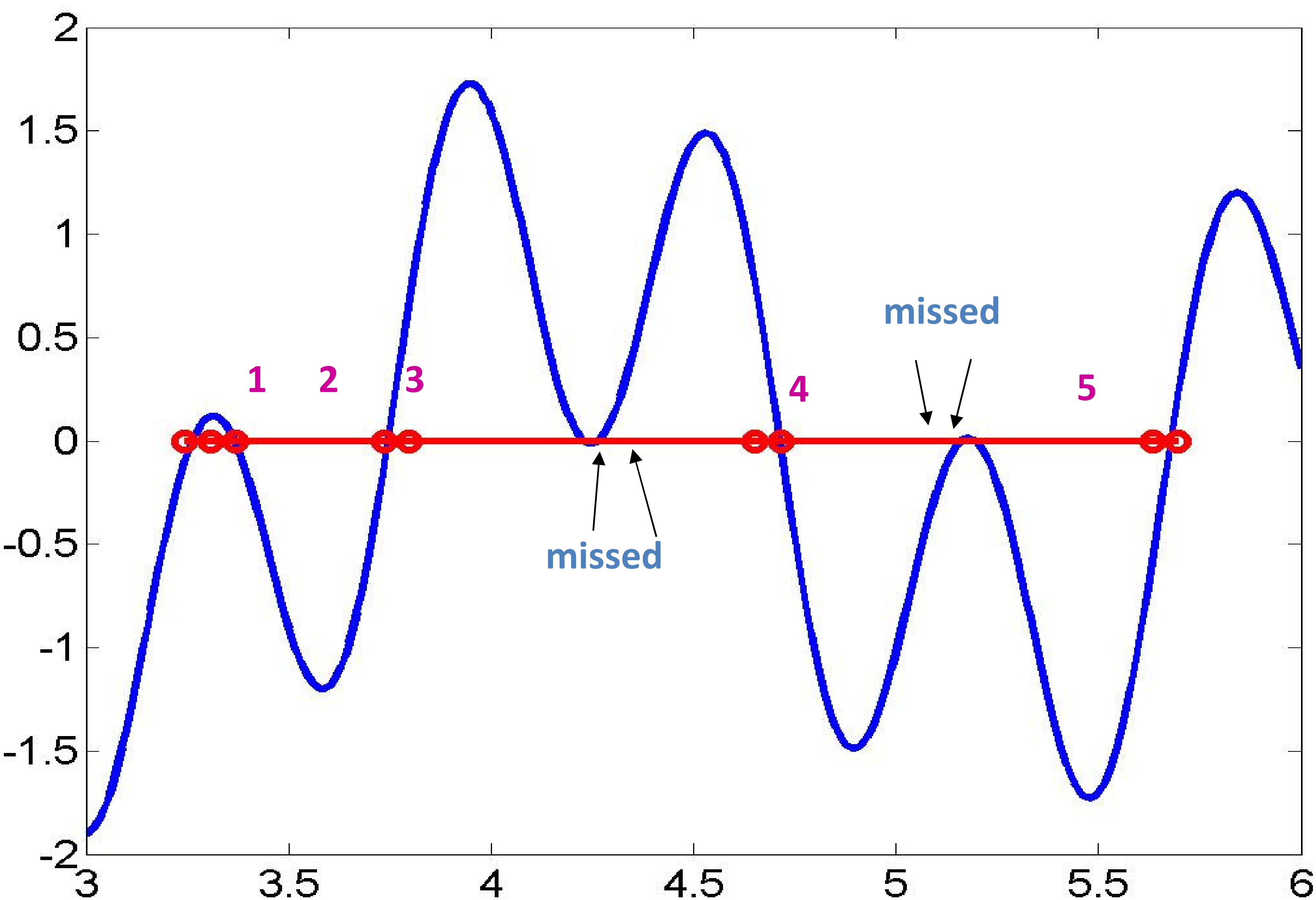
```
function xb = incserach(func, xmin, xmax, ns)
% incsearch(func,xmin,xmax,ns):
%  finds brackets for x that contain sign changes of
%  a function on an interval
% input:
%  func = name of function
%  xmin, xmax = end points of interval
%  ns = (optional) number of subintervals along x
%       used to search for brackets
% output:
%  xb(k,1) is the lower bound of the kth sign change
%  xb(k,2) is the upper bound of the kth sign change

if nargin < 4, ns =50; end  % if ns blank set to 50

% Incremental Search
x = linspace(xmin,xmax,ns);
f = feval(func,x);
nb = 0; xb = [];  % xb is null unless sign change detected
for k = 1: length(x)-1
    if sign(f(k)) ~= sign(f(k+1))  % check for sign change
        nb = nb + 1;
        xb(nb,1) = x(k);
        xb(nb,2) = x(k+1);
    end
end

if isempty(xb)  % display that no brackets were found
    disp('no brackets found')
    disp('check interval or increase ns')
else
    disp('number of brackets:') % display number of brackets
    disp(nb)
end
```

Incremental Search

```
>> xb=incsearch(inline('sin(10*x)+cos(3*x)'),3,6)
number of brackets:
     5
xb =
   3.24489795918367   3.30612244897959
   3.30612244897959   3.36734693877551
   3.73469387755102   3.79591836734694
   4.65306122448980   4.71428571428571
   5.63265306122449   5.69387755102041
>> yb = xb.*0
yb =
     0      0
     0      0
     0      0
     0      0
     0      0
>> x=3:0.01:6; y=sin(10*x)+cos(3*x);
>> plot(x,y,xb,yb,'r-o')
```

Find 5 roots

Use 50 intervals between [3, 6]

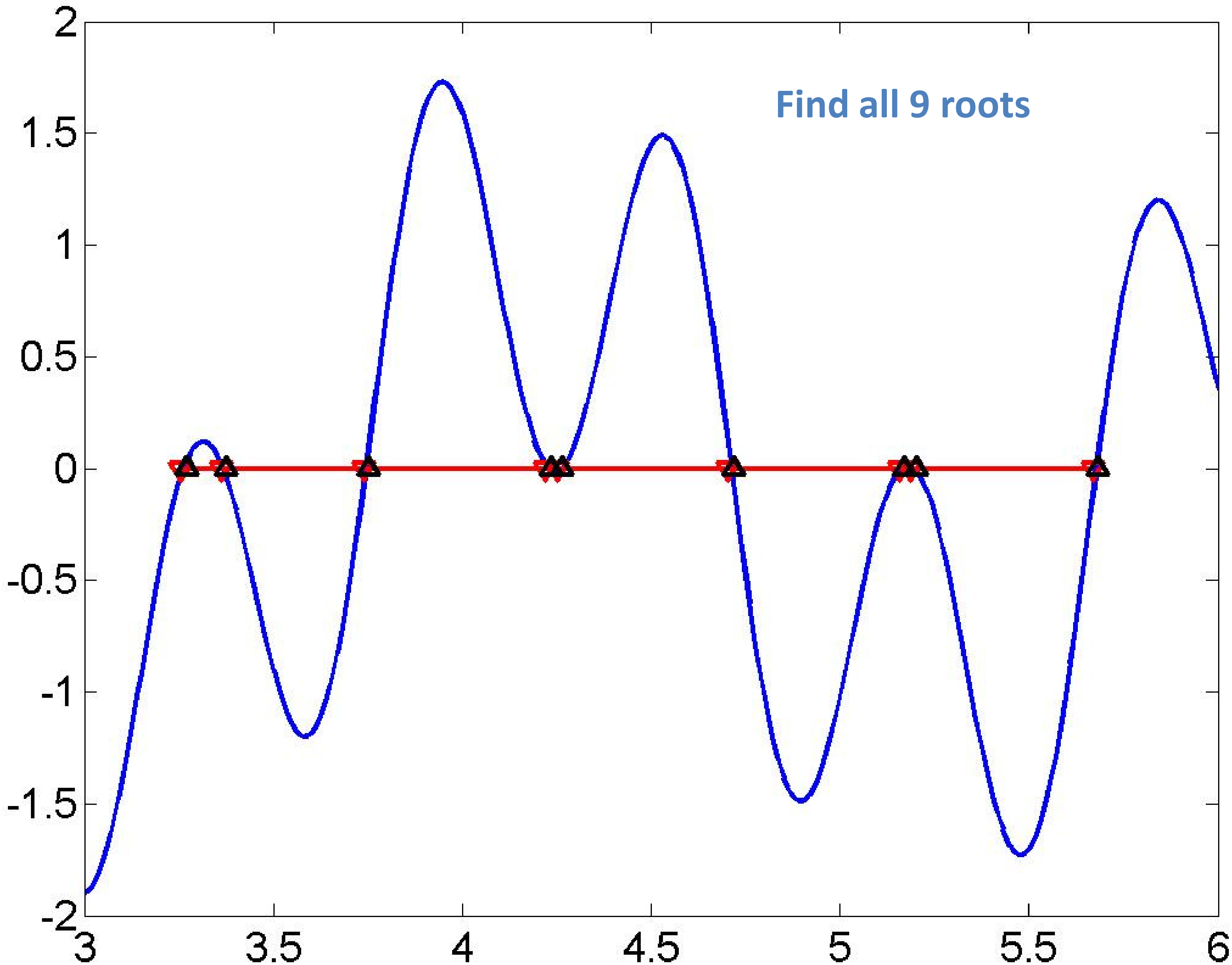


Increase Subintervals to 200

```
>> xb=incsearch(inline('sin(10*x)+cos(3*x)'),3,6,200)
number of brackets:
     9
xb =
   3.25628140703518   3.27135678391960
   3.36180904522613   3.37688442211055
   3.73869346733668   3.75376884422111
   4.22110552763819   4.23618090452261
   4.25125628140704   4.26633165829146
   4.70351758793970   4.71859296482412
   5.15577889447236   5.17085427135678
   5.18592964824121   5.20100502512563
   5.66834170854271   5.68341708542714
>> yb = xb.*0;
>> H = plot(x,y,xb(:,1),yb(:,1),'r-v',xb(:,2),yb(:,2),'k^');
>> set(H,'LineWidth',2,'MarkerSize',8)
```

Find all 9 roots!

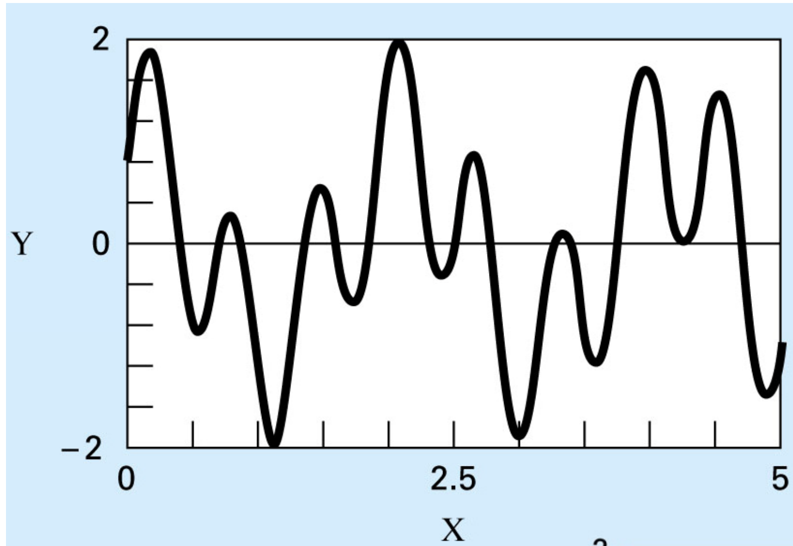
Incremental Search

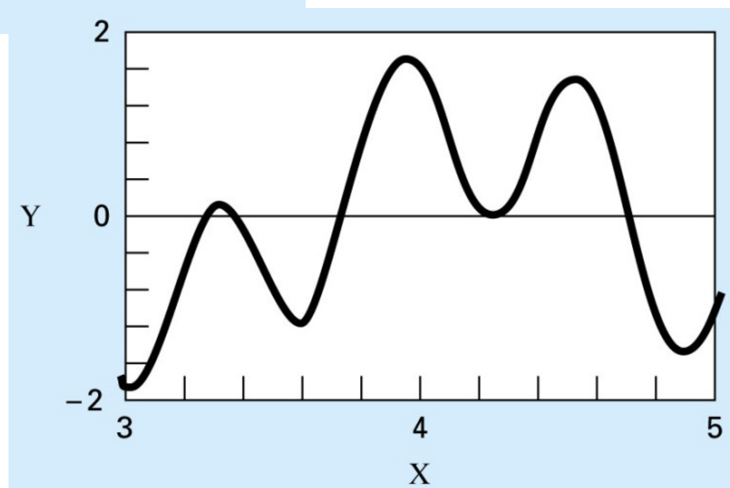


Graphical Approach

- **Make a plot of the function f(x) and observe where it crosses the x-axis, i.e. f(x) = 0**
- **Not very practical but can be used to obtain rough estimates for roots**
- **These estimates can be used as initial guesses for numerical methods.**

Using MATLAB, plot *f(x)=sin(10x)+cos(3x)*





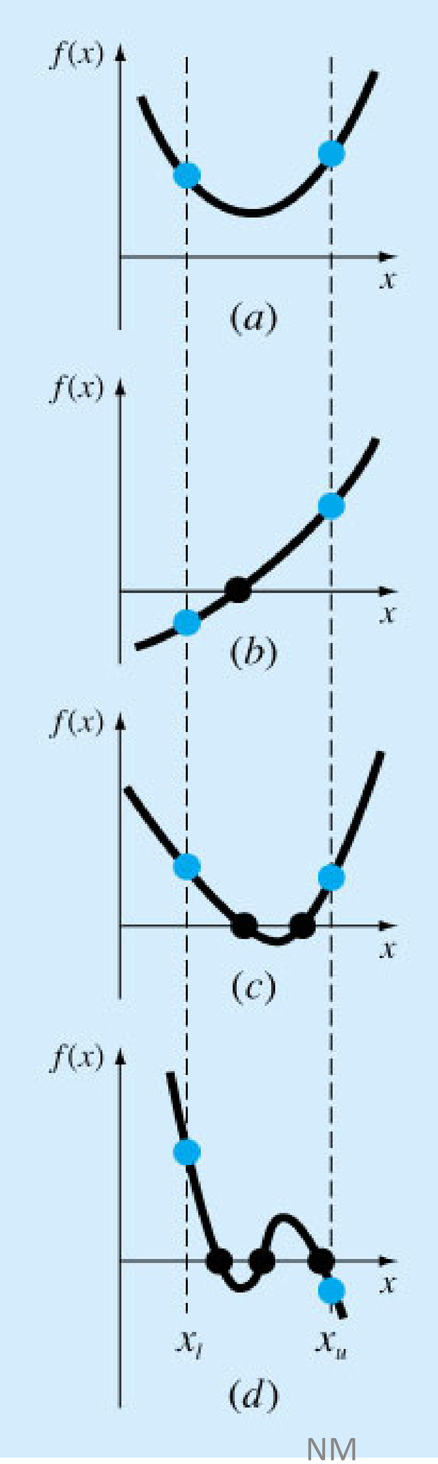
Two distinct roots between

x= 4.2 and 4.3

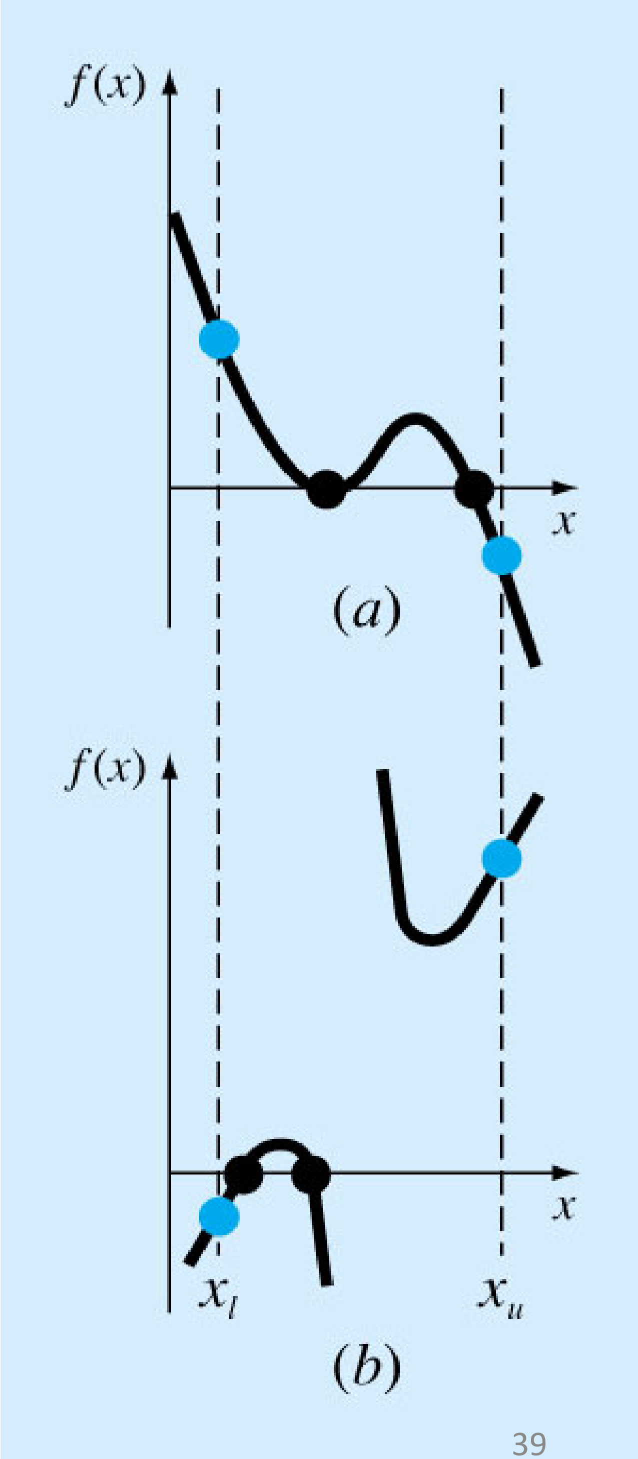
need to be careful

Bracketing:

Odd and even number of roots



exceptions



Bisection Method

$$f(x) = ax^n + bx^{n-1} + cx^{n-2} + dx^{n-3} + \ldots + yx^1 + zx^0 = 0 \quad \Rightarrow \quad x = ?$$

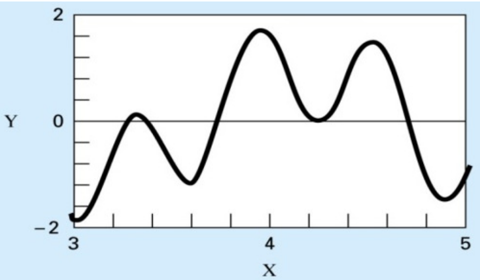
Step 1: Choose lower x_l and upper x_u guesses for the root such that the function changes sign over the interval. This can be checked by ensuring that $f(x_l)f(x_u) < 0$.

$f(x_l) \,\&\, f(x_u)$

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

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

$f(x_r)$



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.

Relative error estimate : $\quad \varepsilon = \dfrac{\left|x_r^{new} - x_r^{old}\right|}{\left|x_r^{new}\right|} 100\%$

Termination criteria: $\varepsilon < \varepsilon_{tol}$ OR *Max.Iteration* is reached

MATLAB code

Bisection Method

- **Minimize function evaluations in the code.**

  Why?

- **Because they are costly (takes more time)**

```
% Bisection Method - simple
% function f(x) = exp(-x) - x = 0   sample call:  bisection(-2, 4, 0.001,500)

function  root = bisection(xl, xu, es, imax);

if ((exp(-xl) - xl)*(exp(-xu) - xu))>0       % if guesses do not bracket, exit
    disp('no bracket')
    return
end

for i=1:1:imax

   xr=(xu+xl)/2;                    % compute the midpoint   xr
   ea = abs((xu-xl)/xl);            % approx. relative error

   test= (exp(-xl) - xl) * (exp(-xr) - xr);    % compute    f(xl)*f(xr)

   if (test < 0)    xu=xr;
   else    xl=xr;
   end

   if (test == 0) ea=0;   end
   if (ea < es)  break;   end

end

s=sprintf('\n  Root= %f    #Iterations = %d \n', xr,i); disp(s);
```

How Many Iterations will It Take?

- Length of the first Interval $L_o = x_u - x_l$
- After 1 iteration $L_1 = L_o/2$
- After 2 iterations $L_2 = L_o/4$

  ….. …..
- After k iterations $L_k = L_o/2^k$

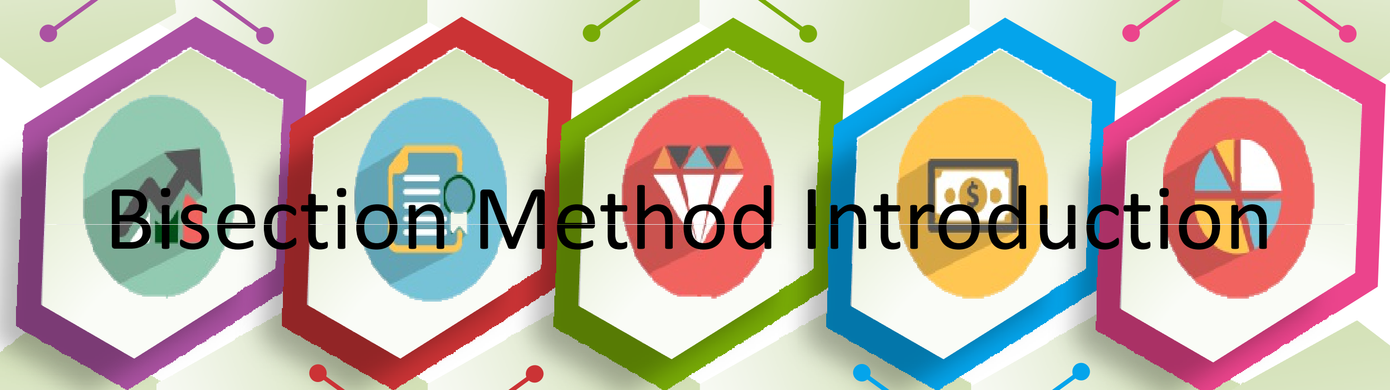
- Then one can write:

$$\left|\frac{L_k}{x_l}\right| \le error\_tolerance$$

$$\left|\frac{L_0}{2^k}\right| \le \left|x_l * \varepsilon_{es}\right|$$

$$2^k \ge \left|\frac{L_0}{x_l * \varepsilon_{es}}\right| \quad \Rightarrow \quad \boxed{k \ge \log_2\left(\left|\frac{L_0}{x_l * \varepsilon_{es}}\right|\right)}$$

Bisection Method Introduction



The **Bisection method** is one of the simplest methods to find a zero of a nonlinear function.

One needs an initial interval that is known to contain a zero of the function.

The procedure is repeated until the desired interval size is obtained.

Interval halving Or Mid - Point Or, **binary search method,** Or **dichotomy method**.

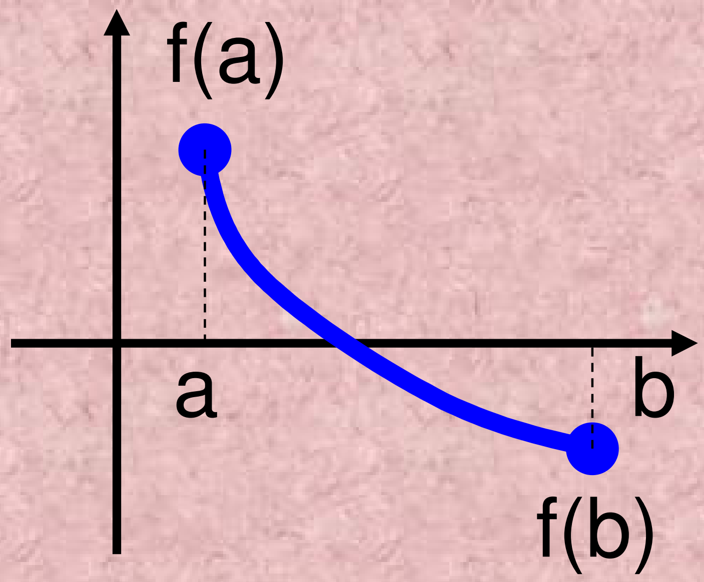
The method systematically reduces the interval. It does this by dividing the interval into two equal parts, performs a simple test and based on the result of the test, half of the interval is thrown away.

Intermediate Value Theorem

- Let f(x) be defined on the interval [a,b].

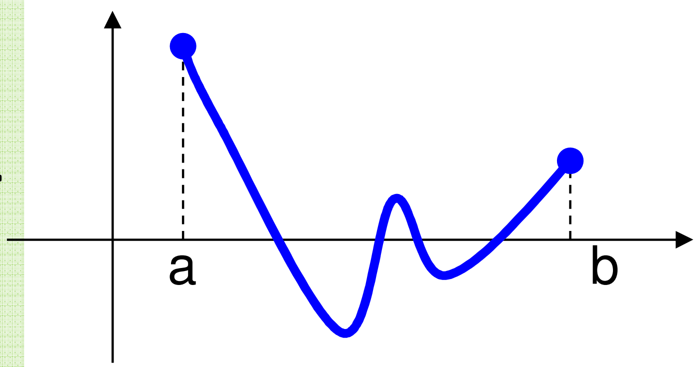
- <u>Intermediate value theorem:</u>

if a function is <u>continuous</u> and f(a) and f(b) have <u>different signs</u> then the function has at least one zero in the interval [a,b].



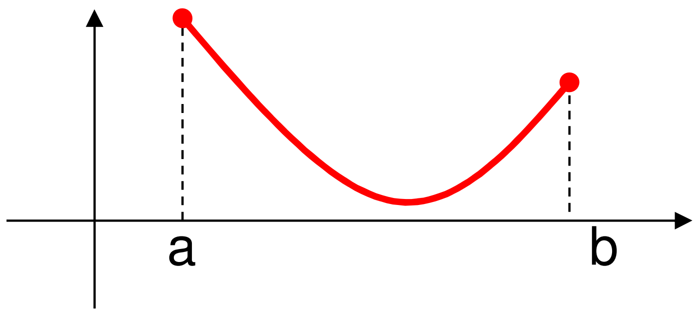
Examples

- If f(a) and f(b) have the same sign, the function may have an even number of real zeros or no real zeros in the interval [a, b].



The function has four real zeros

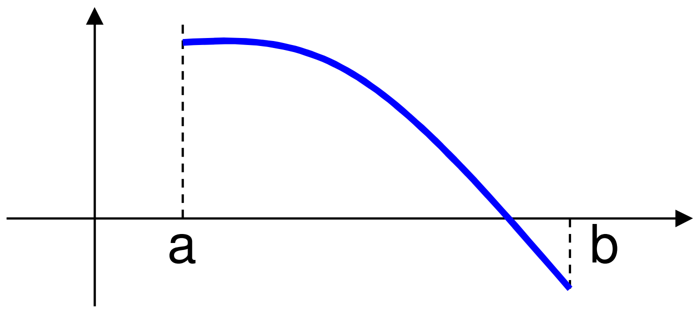
- Bisection method can not be used in these cases.



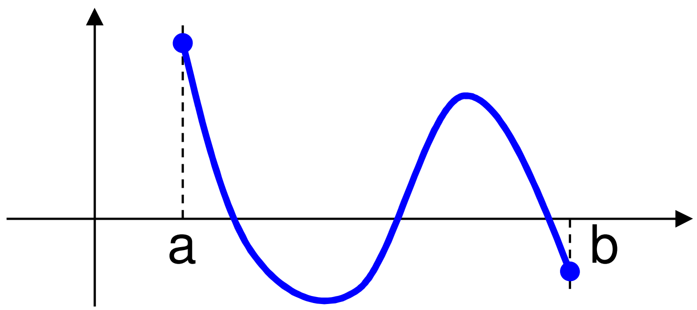
The function has no real zeros

Two More Examples

- If f(a) and f(b) have different signs, the function has at least one real zero.
- It can be used to find one of the zeros.



The function has one real zero



The function has three real zeros

Bisection Method

If the function is continuous on [a,b] and f(a) and f(b) have different signs, Bisection method obtains a new interval that is half of the current interval and the sign of the function at the end points of the interval are different.

This allows us to repeat the Bisection procedure to further reduce the size of the interval.

Bisection Method

Assumptions:

Given an interval [a,b]

f(x) is continuous on [a,b]

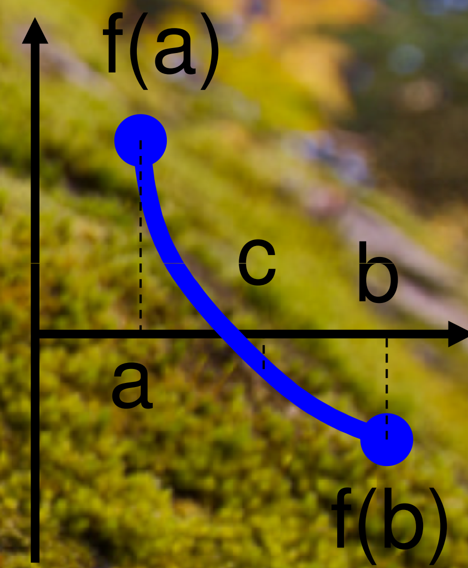
f(a) and f(b) have opposite signs.

These assumptions ensure the existence of at least one zero in the interval [a,b] and the bisection method can be used to obtain a smaller interval that contains the zero.

Bisection Algorithm

Assumptions:

- f(x) is continuous on [a,b]
- f(a) f(b) < 0



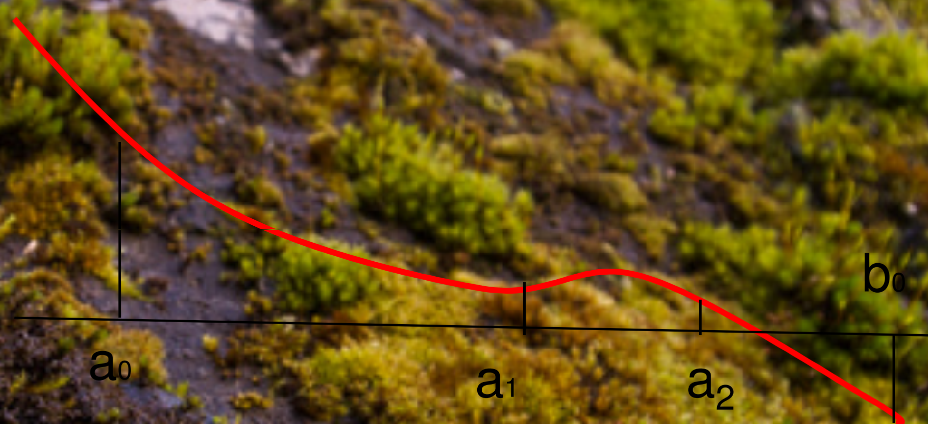
Algorithm:

Loop

1. Compute the mid point c=(a+b)/2
2. Evaluate f(c)
3. If f(a) f(c) < 0 then new interval [a, c]
   If f(a) f(c) > 0 then new interval [c, b]

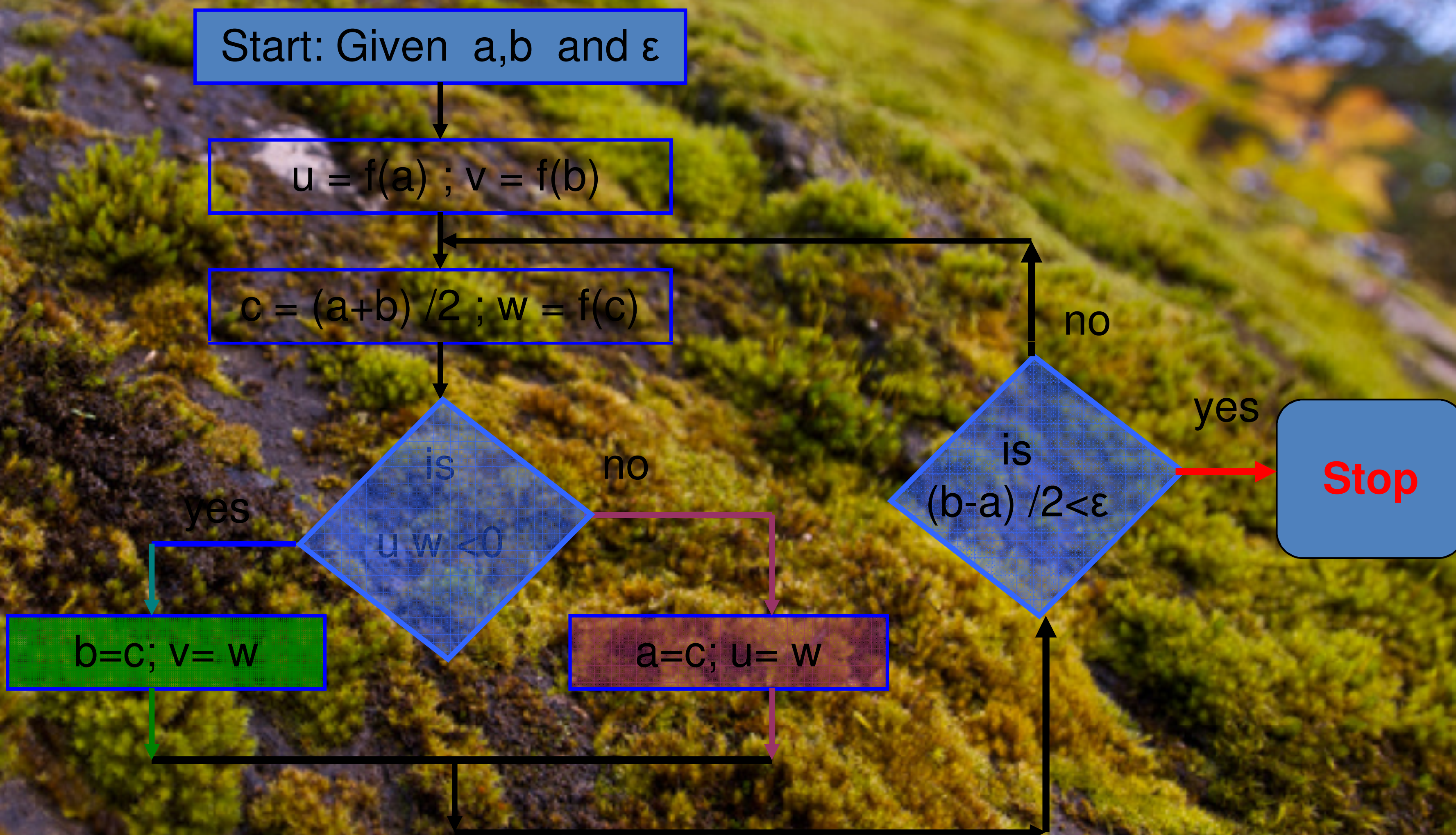
End loop

Bisection Method

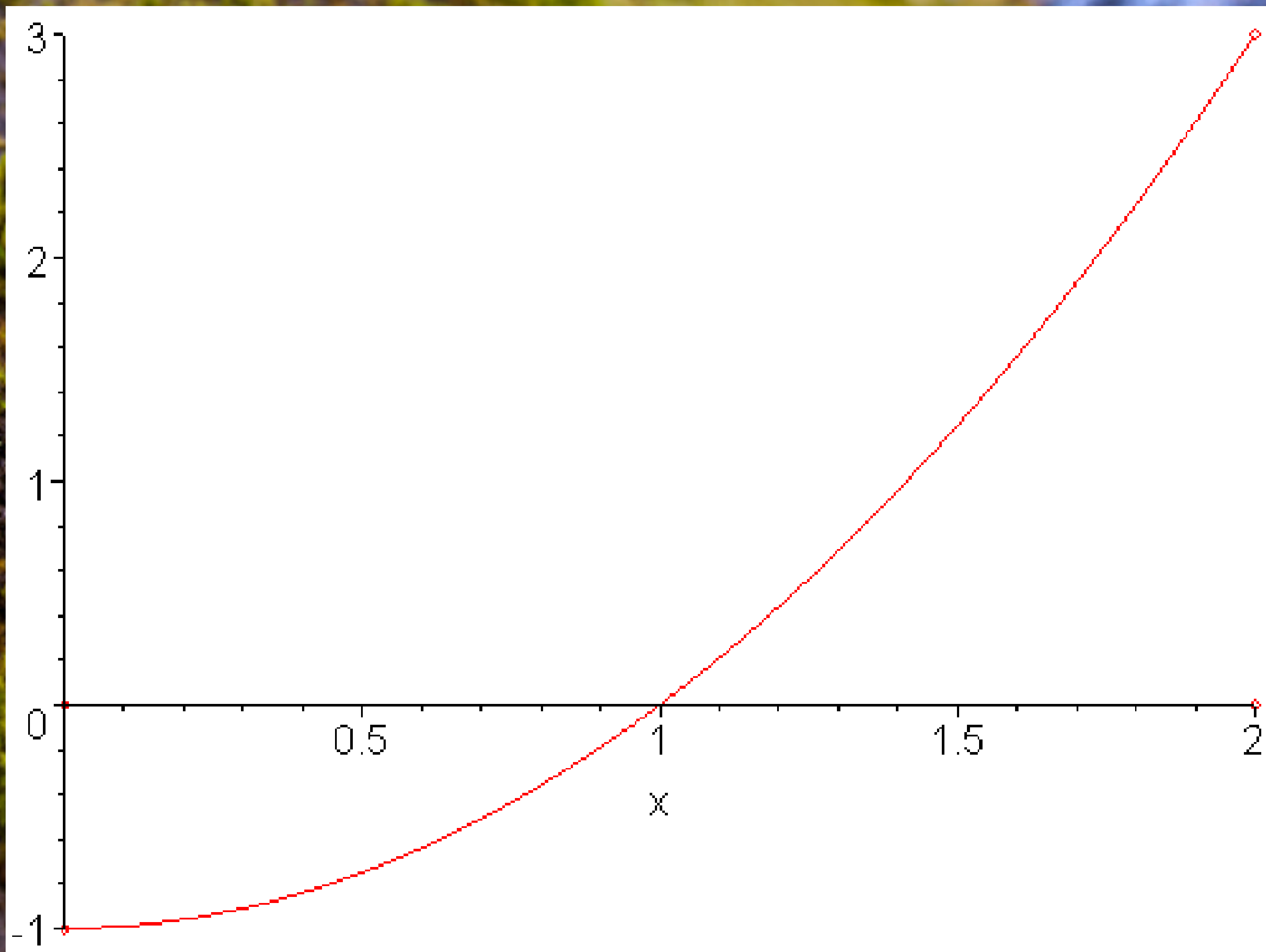


Example

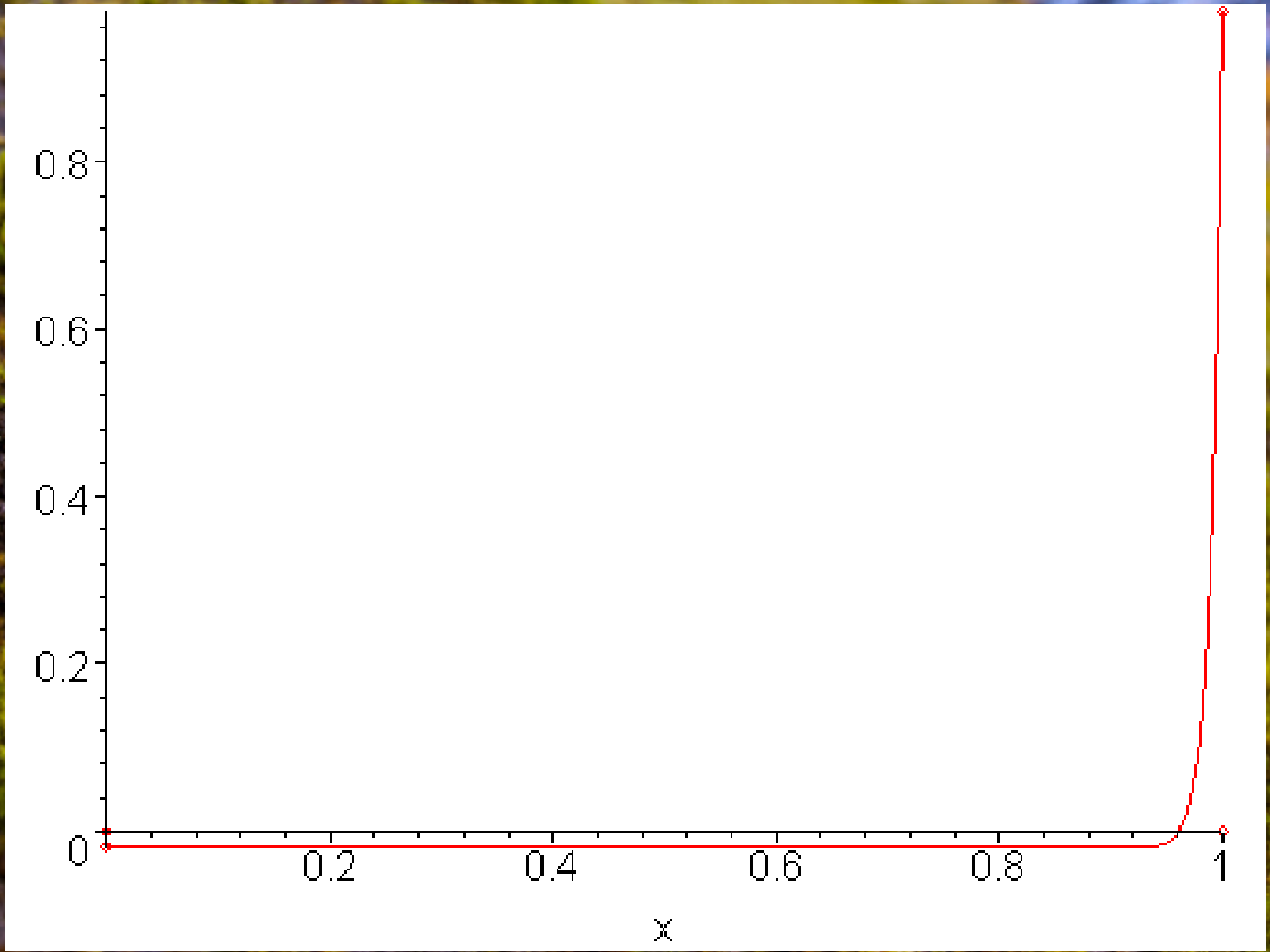
Flow Chart of Bisection Method



Halting Conditions



Effect of Non-linear Functions



Bracketing Methods

- **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**

f(x) = 0 xr is a single root
then f(xl)*f(xu) always < 0
if xl < xr and xu > xr.

Example 1a

Can you use Bisection method to find a zero of :

$f(x) = x^3 - 3x + 1$ in the interval [0,2]?

Answer:

$f(x)$ is continuous on [0,2]

and $f(0) * f(2) = (1)(3) = 3 > 0$

$\Rightarrow$ Assumptions are not satisfied

$\Rightarrow$ Bisection method can not be used

Example 1b

Can you use Bisection method to find a zero of :

$f(x) = x^3 - 3x + 1$ in the interval [0,1]?

Answer:

$f(x)$ is continuous on $[0,1]$

and $f(0) * f(1) = (1)(-1) = -1 < 0$

$\Rightarrow$ Assumptions are satisfied

$\Rightarrow$ Bisection method can be used

Bisection method obtains an interval that is guaranteed to contain a zero of the function.

Best Estimate and Error Level

Questions:

What is the best estimate of the zero of ***f(x)***?

What is the error level in the obtained estimate?

The best estimate of the zero of the function ***f(x)*** after the first iteration of the Bisection method is the mid point of the initial interval:

$$Estimate\ of\ the\ zero:\quad r = \frac{b+a}{2}$$

$$Error \le \frac{b-a}{2}$$

Stopping Criteria

Two common stopping criteria

1. Stop after a fixed number of iterations
2. Stop when the absolute error is less than a specified value

How are these criteria related?

Stopping Criteria

c_n : is the midpoint of the interval at the n[th] iteration

(c_n is usually used as the estimate of the root).

r : is the zero of the function.

After n iterations :

$$|error| = |r - c_n| \le E_a^n = \frac{b-a}{2^n} = \frac{\Delta x^0}{2^n}$$

$$k \ge \log_2\left(\left|\frac{L_0}{x_r * \varepsilon_{es}}\right|\right)$$

Convergence Analysis

Given $f(x)$, a, b, *and* ε

How many iterations are needed such that: $|x - r| \leq \varepsilon$

where r is the zero of $f(x)$ and x is the bisection estimate (i.e., $x = c_k$)?

$$n \geq \frac{\log(b-a)-\log(\varepsilon)}{\log(2)}$$

$$k \geq \log_2\left(\left|\frac{L_0}{x_r * \varepsilon_{es}}\right|\right)$$

Convergence Analysis – Alternative Form

$$n \geq \frac{\log(b-a) - \log(\varepsilon)}{\log(2)}$$

$$n \geq \log_2\left(\frac{\text{width of initial interval}}{\text{desired error}}\right) = \log_2\left(\frac{b-a}{\varepsilon}\right)$$

Example 2

$a = 6, \ b = 7, \ \varepsilon = 0.0005$

How many iterations are needed such that: $\left| x - r \right| \leq \varepsilon$?

$$n \geq \frac{\log(b-a) - \log(\varepsilon)}{\log(2)} = \frac{\log(1) - \log(0.0005)}{\log(2)} = 10.9658$$

$$\Rightarrow n \geq 11$$

Example 2

- Use Bisection method to find a root of the equation $x = \cos(x)$ with absolute error <0.02

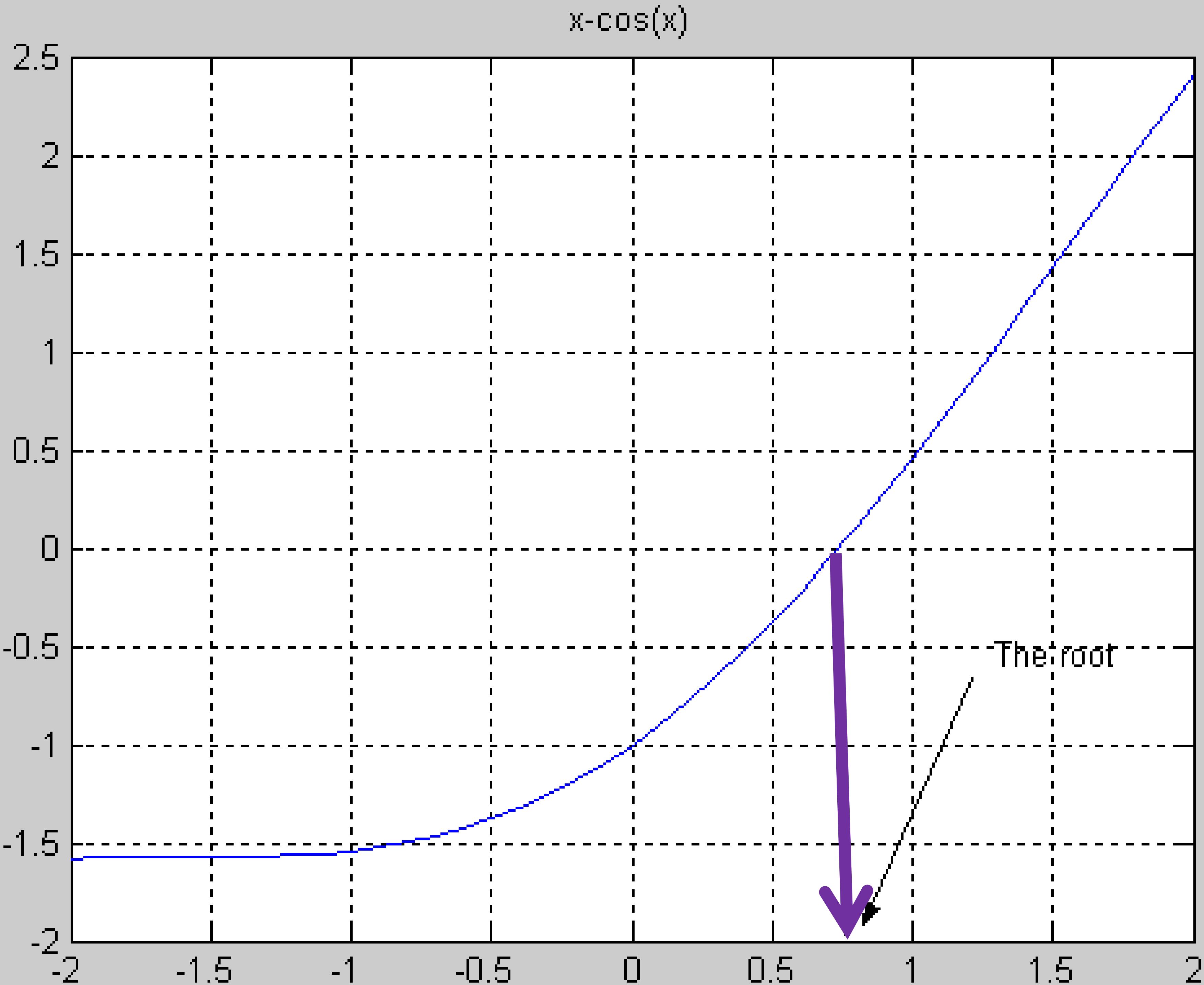
  (assume the initial interval [0.5, 0.9])

Question 1: What is *f (x) ?*

Question 2: Are the assumptions satisfied ?

Question 3: How many iterations are needed ?

Question 4: How to compute the new estimate ?


x-cos(x)
The root
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

Bisection Method

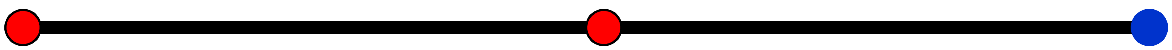
$f(x) = x - \cos(x)$

Initial Interval

[0.5, 0.9]

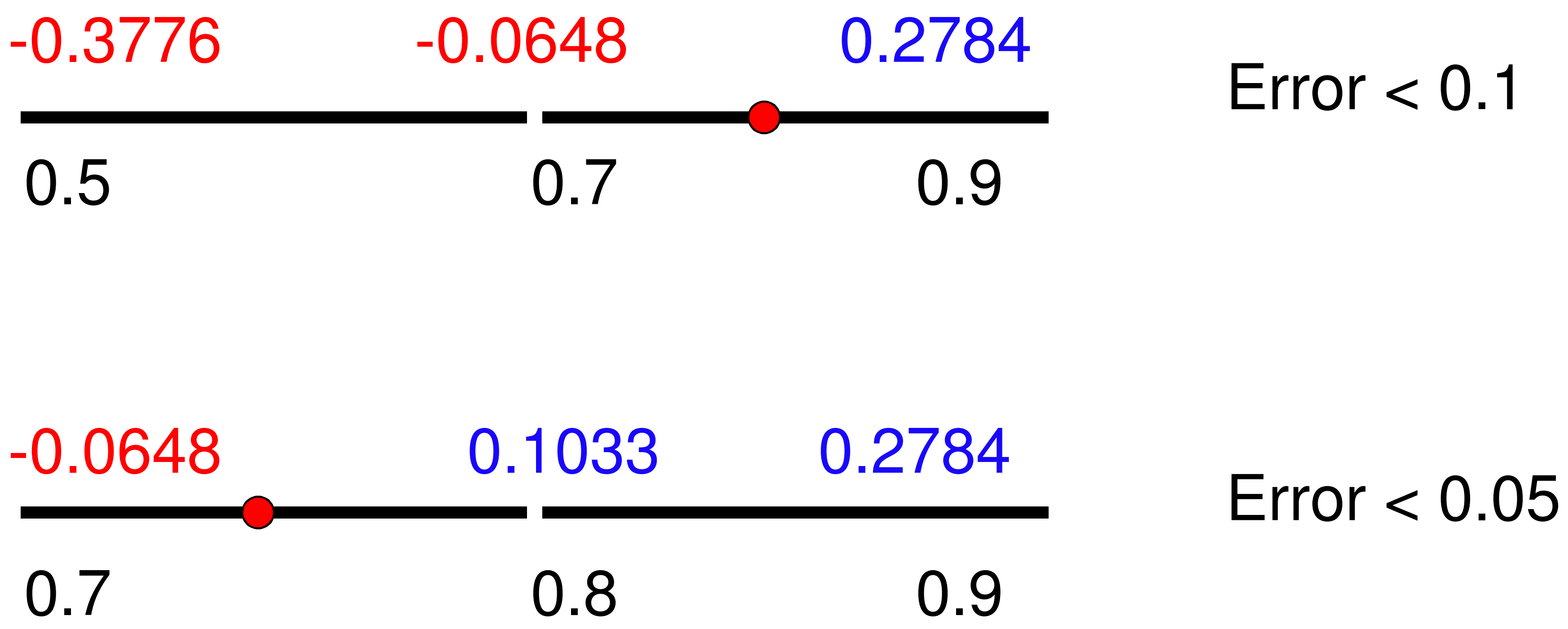
$f(a)=-0.3776$ $f(b)=0.2784$

Error < 0.2

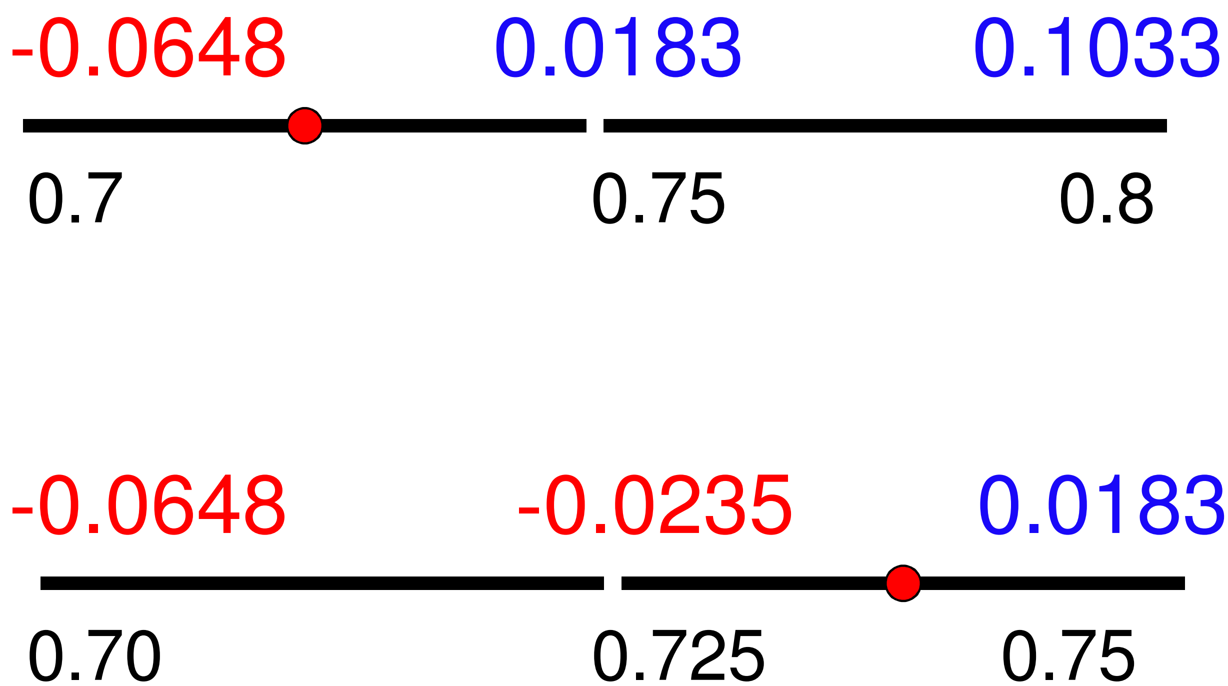


$a=0.5$ $c=0.7$ $b=0.9$

Bisection Method



Bisection Method



Error < 0.025

Error < .0125

Summary

- Initial interval containing the root: [0.5,0.9]

- After 5 iterations:
  - Interval containing the root: [0.725, 0.75]
  - Best estimate of the root is 0.7375
  - | Error | < 0.0125

A Matlab Program of Bisection Method

```
a=.5; b=.9;
 u=a-cos(a);
 v=b-cos(b);
     for i=1:5
           c=(a+b)/2
           fc=c-cos(c)
           if u*fc<0
             b=c ; v=fc;
           else
             a=c; u=fc;
           end
     end
```

```
c =
    0.7000
fc =
   -0.0648
c =
    0.8000
fc =
    0.1033
c =
    0.7500
fc =
    0.0183
c =
    0.7250
fc =
  -0.0235
```

Example 3

Find the root of:

$$f(x) = x^3 - 3x + 1 \quad \text{in the} \quad \text{interval} \quad : [0,1]$$

* $f(x)$ is continuous

* $f(0) = 1, \quad f(1) = -1 \Rightarrow \quad f(a)\, f(b) < 0$

$\Rightarrow$ Bisection method can be used to find the root

$f(x) = x^3 - 3x + 1$ in the interval $:[0,1]$

Example 3

Iteration	a	b	c= $\frac{(a+b)}{2}$	f(c)	$\frac{(b-a)}{2}$
1	0	1	0.5	-0.375	0.5
2	0	0.5	0.25	0.266	0.25
3	0.25	0.5	.375	-7.23E-3	0.125
4	0.25	0.375	0.3125	9.30E-2	0.0625
5	0.3125	0.375	0.34375	9.37E-3	0.03125