Numerical Methods

Roots of Non-linear equations

Contents

Non-linear equations

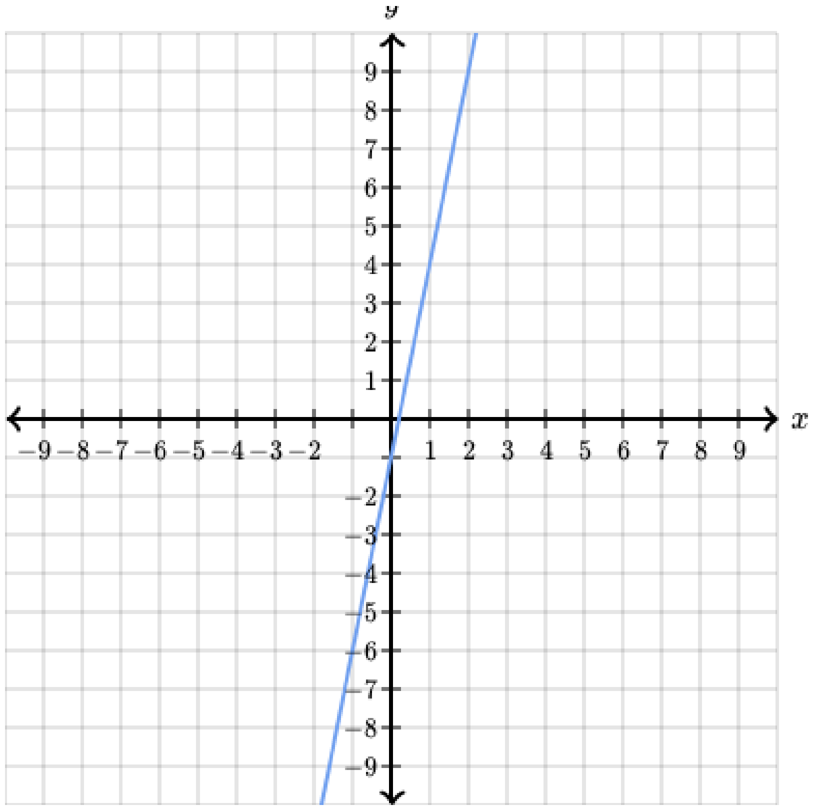
Does the equation define y as a linear function of x?

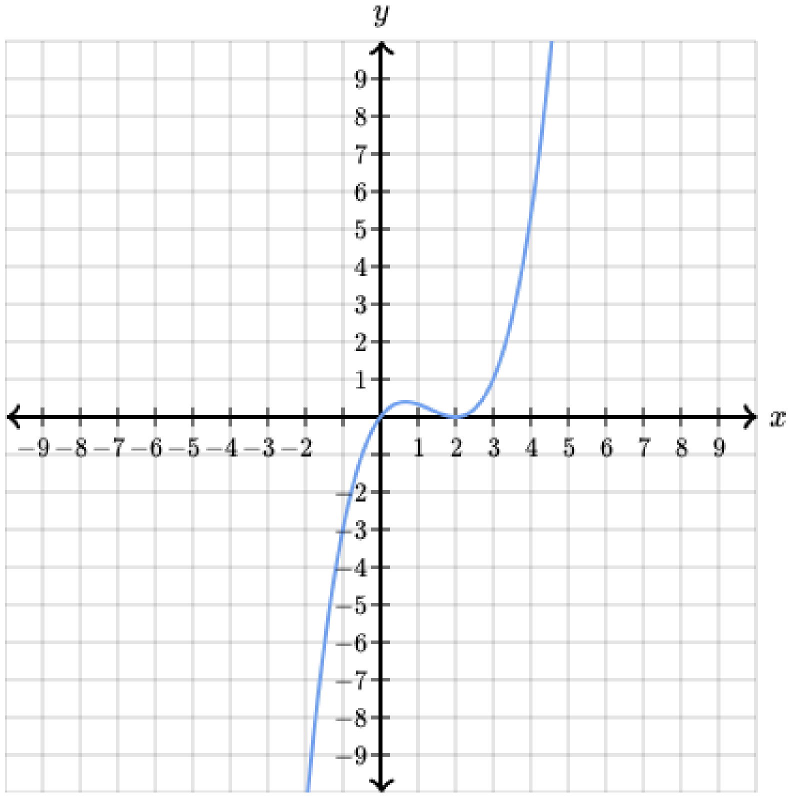
$$y + 4 = 4(x + 1)$$

$$y = x^3 + 1$$

Non-linear equations

Does the graph shown below represent y as a linear function of x?





Non-linear equations

- Linear equations (first degree equation)
  - Algebraic equation in which each term is either a constant or the product of a constant and (the first power of) a single variable
  - *Ex:* $ax + b = 0$, $ax + by + cz + d = 0$ …
- Non-linear equations
  - Equations with exponents greater than one
  - *Ex:* $ax^2 + bx + c = 0$, $y^2 + dy/dx = 0$, …

→ HOW TO SOLVE ?

Roots of Non-linear equations

- Bracketing methods
  - Bisection
  - False position
- Open methods
  - Fixed-point iteration
  - Newton-Raphson
  - Secant methods

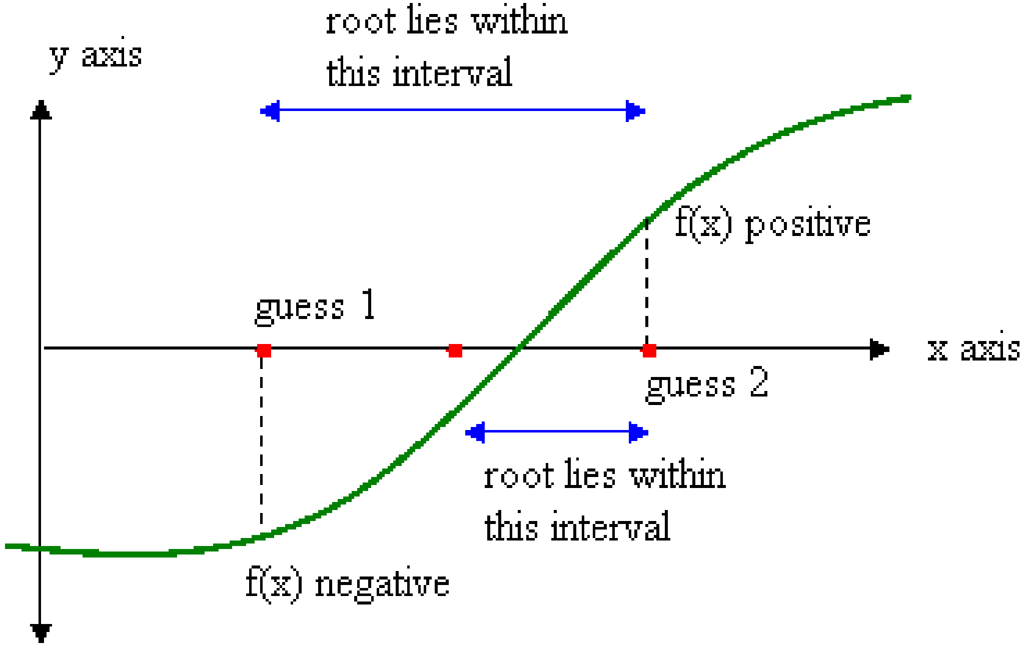
Bisection method

- Objectives: the primary objective is to find the root of a single nonlinear equation
  - Knowing how to solve a roots problem with the bisection method
  - Knowing how to estimate the error of bisection

Example: Find the root of a polynomial $f(x) = x^3 - x - 2$

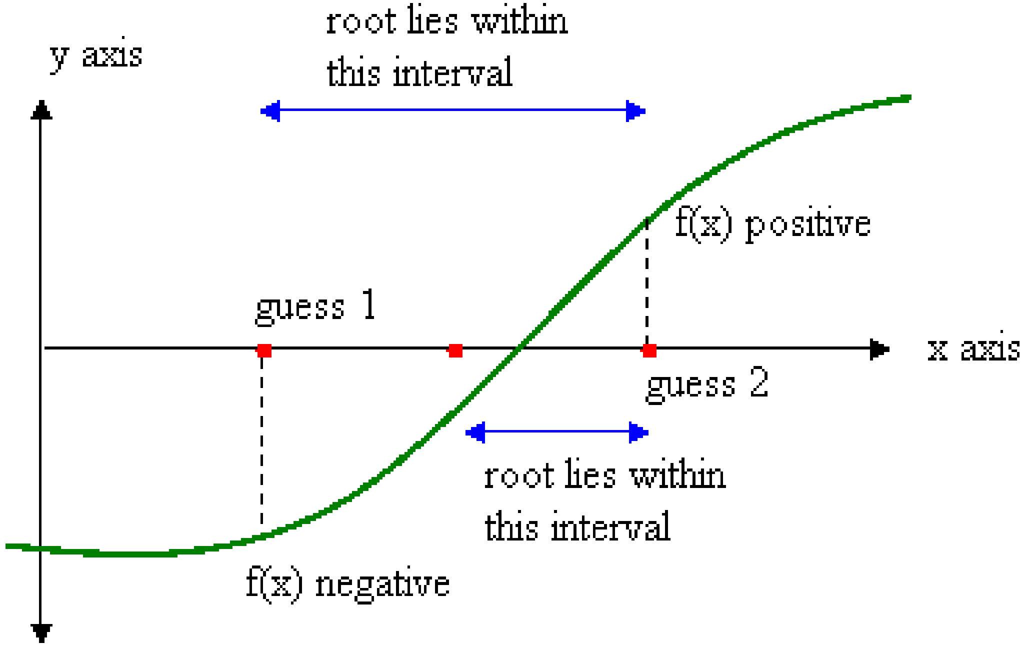
Bisection method

- these are based on two initial guesses that "bracket" the root, are on either side of the root

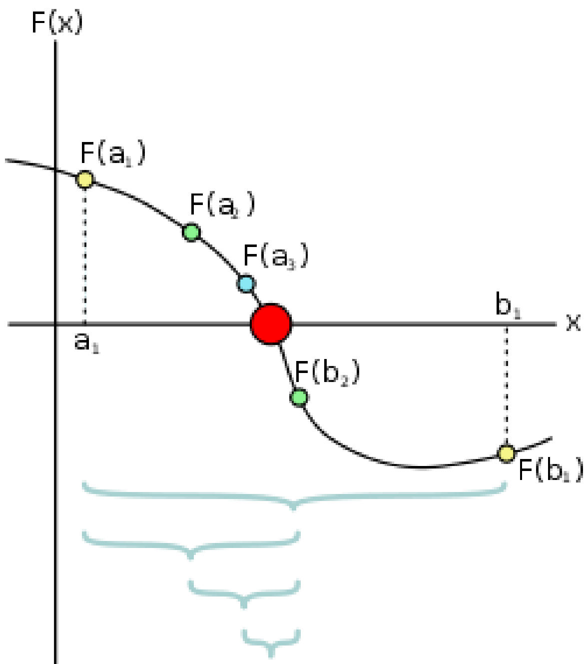


Bisection method

- The bisection method is a root-finding method that repeatedly bisects an interval and then selects a subinterval in which a root must lie for further processing
  - Very simple
  - Robust
  - Relatively slow



- Principle: two initial guesses that "bracket"



A few steps of the bisection method applied over the starting range $[a_1;b_1]$. The bigger red dot is the root of the function

- This is a recursive algorithm because a set of steps are repeated with the previous answer put in the next repetition.
- Each repetition is called an iteration.

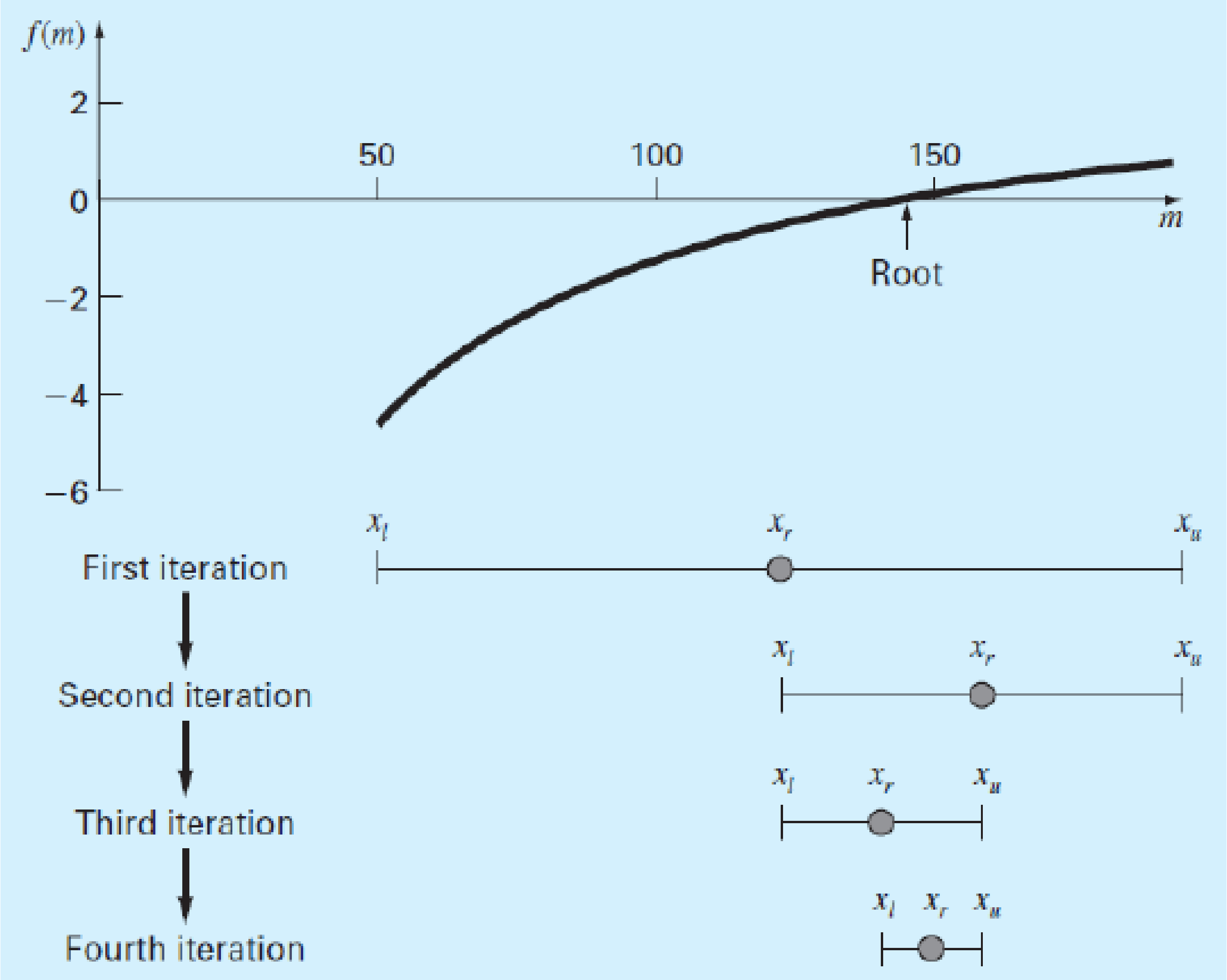
Bisection method

- **Steps**

a continuous function f, an interval $[a, b]$, and the function values $f(a)$ and $f(b)$. The function values are of opposite sign (there is at least one zero crossing within the interval)

1. Calculate the midpoint of the interval, $c = 0.5 * (a + b)$
2. Calculate the function value at the midpoint, $f(c)$
3. If convergence is satisfactory (that is, $a - c$ is sufficiently small, or $f(c)$ is sufficiently small), return c and stop iterating
4. Examine the sign of $f(c)$ and replace either $(a, f(a))$ or $(b, f(b))$ with $(c, f(c))$ so that there is a zero crossing within the new interval

Bisection method



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

First iteration

$$x_r = (x_l + x_u)/2$$

$$f(x_l)f(x_r) > 0$$

$$x_r \to x_l$$

Second iteration

$$x_r = (x_l + x_u)/2$$

$$f(x_r)f(x_u) > 0$$

$$x_r \to x_u$$

Third iteration

$$x_r = (x_l + x_u)/2$$

$$f(x_l)f(x_r) > 0$$

$$x_r \to x_l$$

❖ The error estimator

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

x_r^{new} : the root for the present iteration

x_r^{old}: the root from the previous iteration

When $|\varepsilon_a| \leq \varepsilon_s$, the computation is terminated

ε_s: prespecified stopping criterion (the prespecified acceptable level)

- The true relative error

$$\varepsilon_t = \frac{\text{true error}}{\text{true value}} 100\%$$

where ε_t designates the true percent relative error.

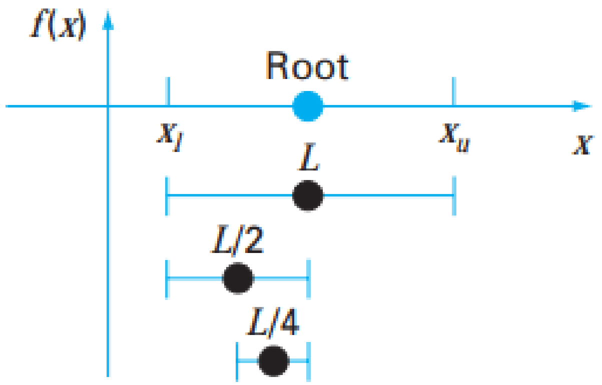
- The relative error

$$\varepsilon_a = \frac{\text{approximate error}}{\text{approximation}} 100\%$$

$$\varepsilon_a = \frac{\text{current approximation} - \text{previous approximation}}{\text{current approximation}} 100\%$$

❖ Summary

Method	Formulation
Bisection	$x_r = \dfrac{x_l + x_u}{2}$ If $f(x_l)f(x_r) < 0,\ x_u = x_r$ $f(x_l)f(x_r) > 0,\ x_l = x_r$

Graphical Interpretation	Errors and Stopping Criteria
Bracketing methods: 	Stopping criterion: $\left\lvert \dfrac{x_r^{new} - x_r^{old}}{x_r^{new}} \right\rvert 100\% \le \epsilon_s$

Case study

Find the root of a polynomial $f(x) = x^3 - x - 2$

Step 1: choose two numbers a and b have to be found such that f(a).f(b) < 0

$$a = 1\ ,\ \ b = 2$$

Step 2: verify this condition f(a).f(b) < 0

$$\left.\begin{aligned} f(1) &= (1)^3 - 1 - 2 = -2 \\ f(2) &= (2)^3 - 2 - 2 = +4 \end{aligned}\right\} \quad f(a).f(b) = -8 < 0 \;\Rightarrow\; \text{Satisfy}$$

Case study

Find the root of a polynomial

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

Step 3: calculate the midpoint

$$c_1 = \frac{2+1}{2} = 1.5$$

Case study

Step 4: calculate the function value at midpoint

$$f(c_1) = (1.5)^3 - 1.5 - 2 = -0.125$$

Step 5: Because $f(c_1)$ is negative, a=1 is replaced with $a=c_1=1.5$ for the next iteration to ensure that f(a) and f(b) have opposite signs

Step 6: Repeat step 3 through 5

$$c_2 = \frac{2+1.5}{2} = 1.75$$

$$f(c_2) = (1.75)^3 - 1.75 - 2 = 1.609 \Rightarrow a = 1.5\ ,\ b = 1.75$$

Case study

Iteration	a_n	b_n	c_n	$f(c_n)$
1	1	2	1.5	−0.125
2	1.5	2	1.75	1.6093750

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

As this continues, the interval between a and b will become increasingly smaller, converging on the root of the function

After 13 iterations, it becomes apparent that there is a convergence to about c =1.521: a root for the polynomial

Example 1

Find the root of a polynomial

- Start with the interval [1,2]
- 5 iterations

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

Example 1

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

a	b	f(a)	f(b)	c = (a + b)/2	f(c)	Update	new b − a
1.0	2.0	-2.0	1.0	1.5	-0.75	a = c	0.5

- Bisection Method Disadvantages
  - The bisection method only finds roots where the function crosses the x axis. It cannot find roots where the function is tangent to the x axis.
  - The bisection method can be fooled by singularities in the function.
  - The bisection method cannot find complex roots of polynomials

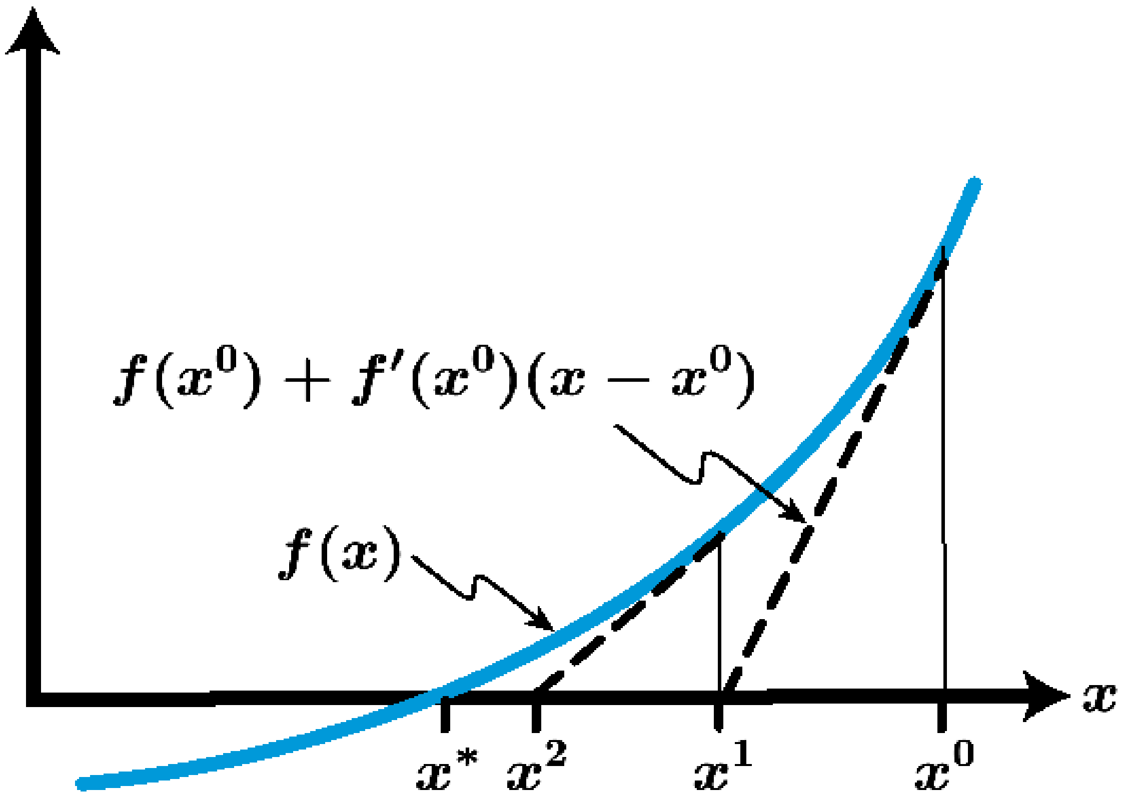
- Bracketing methods
  - Bisection
  - false position
- Open methods
  - Fixed-point iteration
  - Newton-Raphson
  - Secant methods

Open methods

- Objectives: the primary objective is to find the root of a single nonlinear equation
  - Knowing how to solve a roots problem with the Newton-Raphson method and appreciating the concept of quadratic convergence.
  - Learning how to manipulate and determine the roots of polynomials with MATLAB

Newton-Raphson

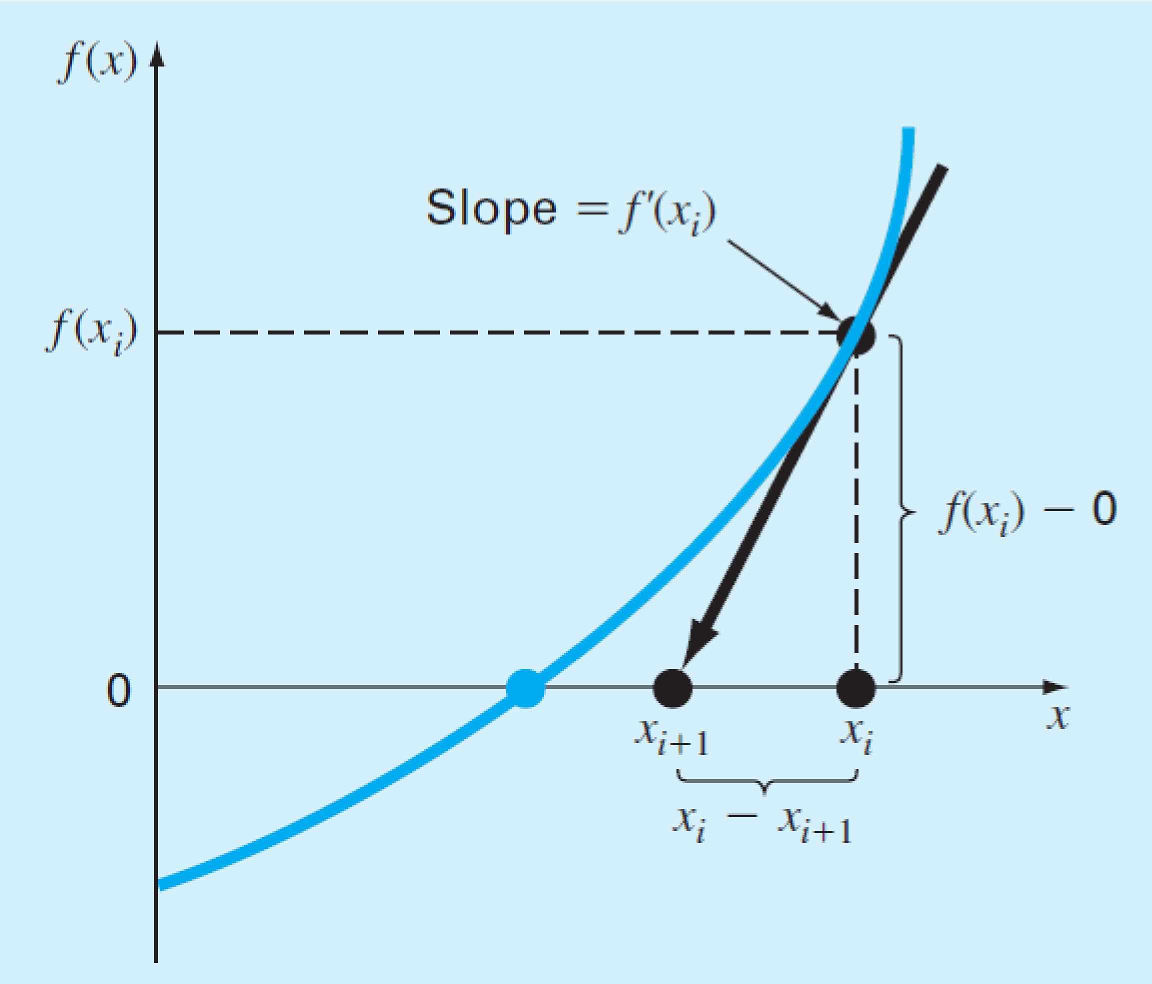
- illustration



only one initial value is required

Newton-Raphson

- Principle



the first derivative at x is equivalent to the slope

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

which can be rearranged to yield

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

which is called the Newton-Raphson formula

Newton-Raphson

- **Steps**

Input: a continuous function f,

1. Evaluate $f'(x)$ symbolically
2. Use an initial guess of the root x_i, to estimate the new value of the root x_{i+1}, as

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

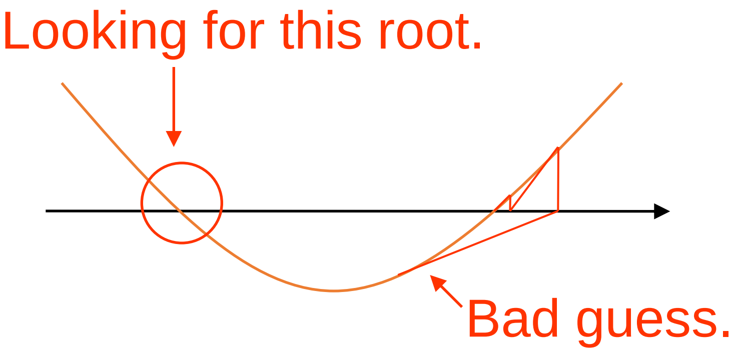
3. Find the relative approximate error

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

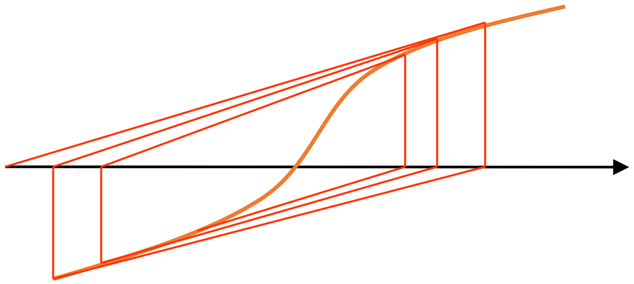
When $|\varepsilon_a| \leq \varepsilon_s$, the computation is terminated

Newton-Raphson

- limitations to Newton’s method



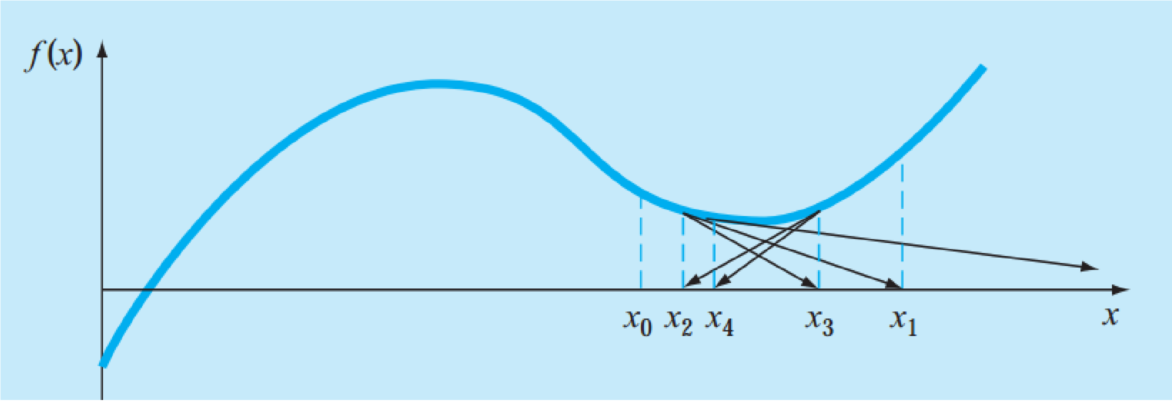
Wrong root found

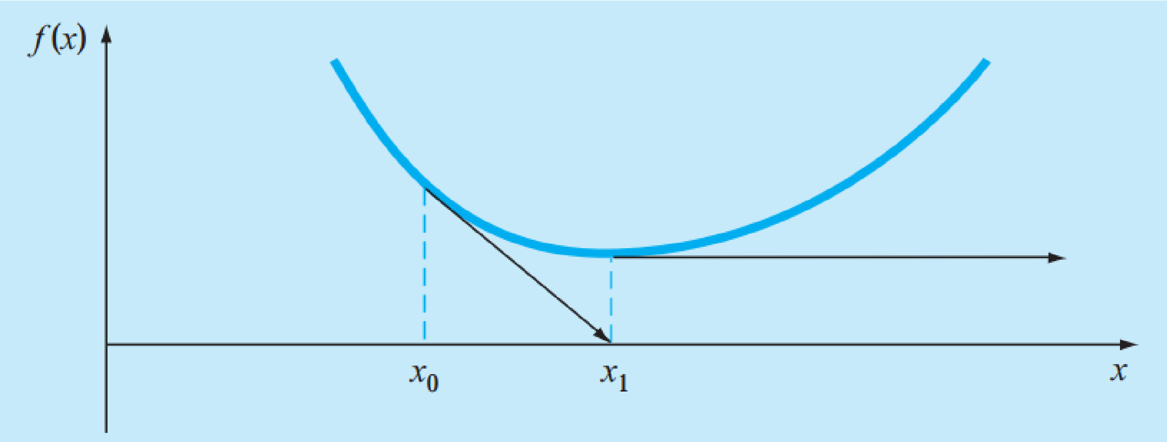


Failure to converge

Newton-Raphson

- limitations to Newton's method





- limitations to Newton's method

Problem Statement. Determine the positive root of $f(x) = x^{10} - 1$ using the Newton-Raphson method and an initial guess of $x = 0.5$.

Solution. The Newton-Raphson formula for this case is

$$x_{i+1} = x_i - \frac{x_i^{10} - 1}{10x_i^9}$$

which can be used to compute

Iteration	x
0	0.5
1	51.65
2	46.485
3	41.8365
4	37.65285
5	33.887565
.	
.	
.	
∞	1.0000000

very slow rate

Case study 1

Use the Newton-Raphson method to estimate the root of

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

employing an initial guess of $x_0 = 0$

Case study 1

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

Step 1: The first derivative of the function can be evaluated as

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

Step 2: which can be substituted along with the original function into

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

to give

$$x_{i+1} = x_i - \frac{e^{-x_i} - x_i}{-e^{-x_i} - 1}$$

Case study 1

Starting with an initial guess of $x_0 = 0$, this iterative equation can be applied to compute

$$x_{i+1} = x_i - \frac{e^{-x_i} - x_i}{-e^{-x_i} - 1}$$

i	x_i	$\|\varepsilon_t\|$, %
0	0	100
1	0.500000000	11.8
2	0.566311003	0.147
3	0.567143165	0.0000220
4	0.567143290	$<10^{-8}$

→ the approach rapidly converges on the true root

Case study 2

Use the Newton-Raphson method to estimate the root of

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

employing an initial guess of $x_0 = 0$

Case study 2

- Solution

The first derivative of the function can be evaluated as

$$\boxed{f'(x) = 3x^2 - 1}$$

which can be substituted along with the original function into

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

to give

$$\boxed{x_{i+1} = x_i - \frac{x_i^3 - x_i - 2}{3x_i^2 - 1}}$$

Case study 2

Starting with an initial guess of $x_0 = 0$, this iterative equation can be applied to compute

$$x_{i+1} = x_i - \frac{x_i^3 - x_i - 2}{3x_i^2 - 1}$$

i	x_i	err
1	-2	100
2	-1,272	57,14
3	-0,5	131
4	-18,1	96,9
5	-12,08	49,87
6	-8,067	49,74
…		
31	1,521	3,20E-09

Case study 3

Starting with an initial guess of $x_0 = 2$

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

$$\boxed{x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}} \quad \Rightarrow \quad x_{n+1} = x_n - \frac{(x_n^3 - x_n - 1)}{(3x_n^2 - 1)}$$

Using a calculator we need: 2, ENTER

Then $$ANS - \frac{(ANS^3 - ANS - 1)}{(3ANS^2 - 1)}$$

X = 1.6180

Matlab-correction

```
%% Quadratic equation
clc
clear all
close all
a=1
b=2
c=-3
if a==0
    [x]=function1(b,c);
else
    [x1 x2]=function2(a,b,c);
    if x1==x2
        fprintf('Double solution:')
    end
    x1
    x2
end
```

```
function [ x1,x2 ] = function2(a,b,c)
delta = b^2-4*a*c;
if delta>0
    x1=(-b+sqrt(delta))/(2*a);
    x2=(-b-sqrt(delta))/(2*a);
else
    if delta<0
        x1=(-b+i*sqrt(-delta))/(2*a);
        x2=(-b-i*sqrt(-delta))/(2*a);
    else
        x1=-b/(2*a);
        x2=-b/(2*a);
    end
end

end
```

```
%% Cubic equation
a=1; b=-6;
c=11; d=-6;
if a==0
    if b~=0
        [x1 x2]=function3(b,c,d)
        if x1==x2
            fprintf('Double solution:')
        end
else
        x=function1(c,d)
    end
else
    [x1 x2 x3]=function3 (a,b,c,d)
end
```

```
function [x1,x2,x3] = function3(a,b,c,d)
    m=(3*a*c-b^2)/(3*a^2);
n=(2*b^3+27*a^2*d-9*a*b*c)/(27*a^3);
    [U V]=function2 (1,n,-(m^3)/27);
if U<0
        u=-(-U)^(1/3);
    else u=U^(1/3);
    end
    if V<0
        v=-(-V)^(1/3);
    else v=V^(1/3);
    end
    x1=u+v-b/(3*a);
[x2 x3]=function2(a,a*x1+b,a*x1^2+b*x1+c);
end
```

```
%% Fibonacci numbers
n=10;
x=zeros(1,n);
x(1)=1;
x(2)=1;
for i=3:n
    x(i)=x(i-1)+x(i-2);
end
if n==1
    x(1)
elseif n==2
    disp([x(1) x(2)])
else x
end
```

Matlab-Bisection method

- INPUT: Function f, endpoint values a, b, tolerance TOL, maximum iterations NMAX

CONDITIONS: a < b, either f(a) < 0 and f(b) > 0 or f(a) > 0 and f(b) < 0

- OUTPUT: value which differs from a root of f(x)=0 by less than TOL

```
N ← 1
While N ≤ NMAX # limit iterations to prevent infinite loop
        c ← (a + b)/2 # new midpoint
        If f(c) = 0 or (b – a)/2 < TOL then # solution found
                Output(c)
                Stop
        EndIf
        N ← N + 1 # increment step counter
        If sign(f(c)) = sign(f(a)) then a ← c else b ← c # new interval
EndWhile
Output("Method failed.") # max number of steps exceeded
```

Algorithm for Newton-Raphson

1. A plotting routine should be included in the program.
2. At the end of the computation, the final root estimate should always be substituted into the original function to compute whether the result is close to zero. This check partially guards against those cases where slow or oscillating convergence may lead to a small value of ε_a while the solution is still far from a root.
3. The program should always include an upper limit on the number of iterations to guard against oscillating, slowly convergent, or divergent solutions that could persist interminably.
4. The program should alert the user and take account of the possibility that $f'(x)$ might equal zero at any time during the computation.

Algorithm for Newton-Raphson

input x, M

$k \leftarrow 0$

$y \leftarrow f(x)$

print k, x, y

for $k = 1, 2, \ldots, M$

$\qquad x \leftarrow x - \frac{y}{f'(x)}$

$\qquad y \leftarrow f(x)$

$\qquad$ **print** k, x, y

end

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

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

```
%% Newton Raphson
x = 0;
x_i = 1;
iter = 0;
eps_s = 10^-8
while abs(x_i-x)> eps_s
    x_i = x;
    x = x - (exp(-x) - x)/(-exp(-x) - 1);
    iter = iter + 1;
    fprintf('Iteration %d: x=%.20f, err=%.20f\n', iter, x, abs((x_i-x)/x)*100);
    pause;
end
```

Exercise

```
%% bisection
f=@(x)(x^3-x-2)
a=1
b=2
[p,i] = bisection(f,a,b)
```

Write the function: bisection(f,a,b) to solve the quadratic equation