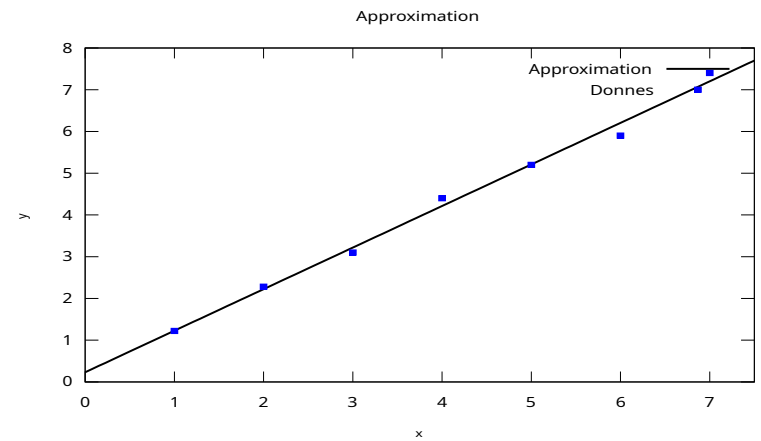
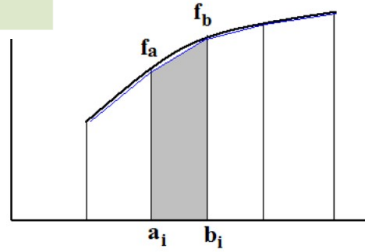
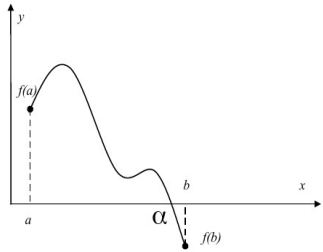
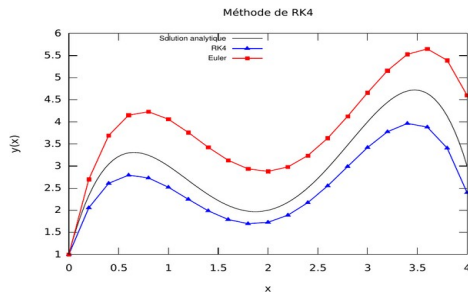


$$L_i(x) = \frac{(-1)^n}{(n-i)! i! h^n} \prod_{j=0, j \neq i}^n (x - x_j)$$



Numerical Analysis

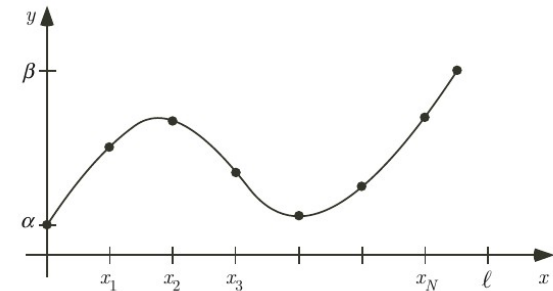
Pr. Righi Haroun



HNSREESD
2026/2027



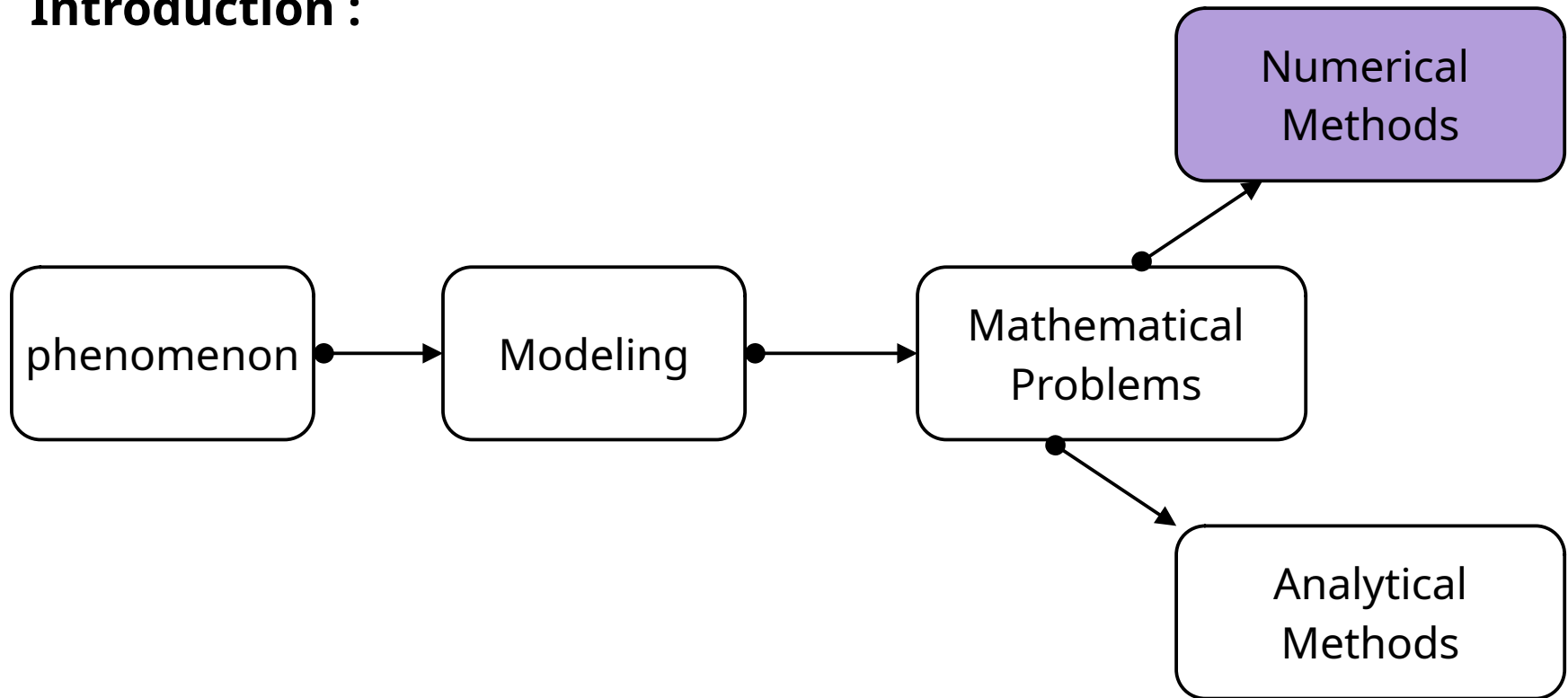
$$x_i^{k+1} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^k - \sum_{j=i+1}^n a_{ij} x_j^k \right)$$



Introduction



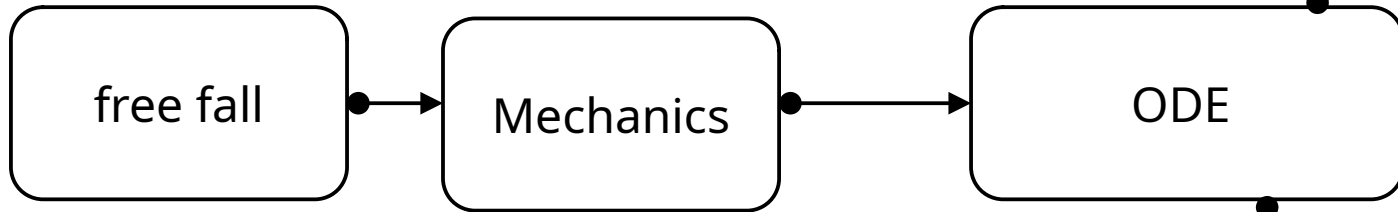
Introduction :



Introduction

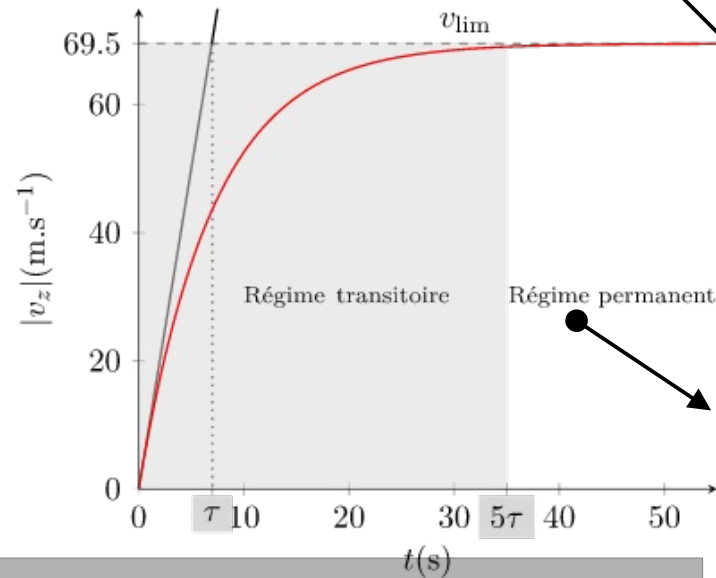
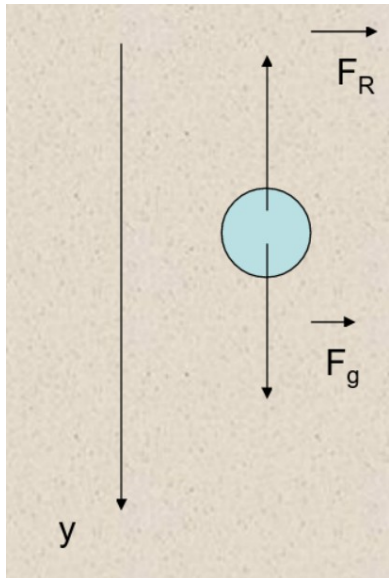


Introduction :



Numerical
Methods

Analytical
Methods

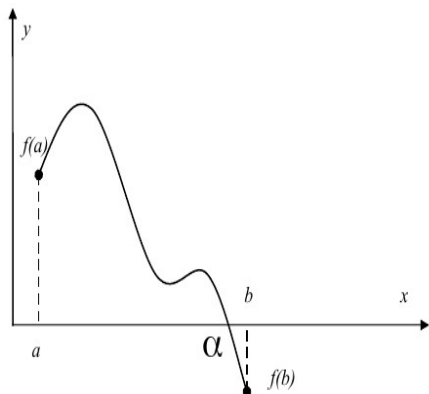


Numerical Analysis 1

Chapter I: Introduction to numerical analysis, Concept of Errors

Chapter II: Nonlinear Equation Solving
(Bisection, Secante, Newton, Fixed Point

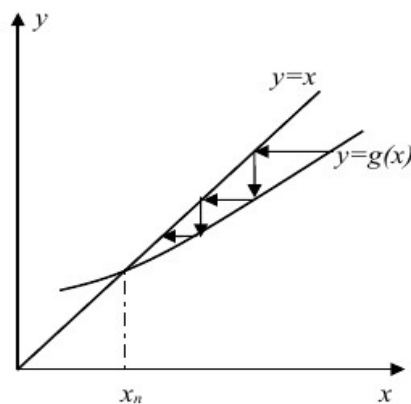
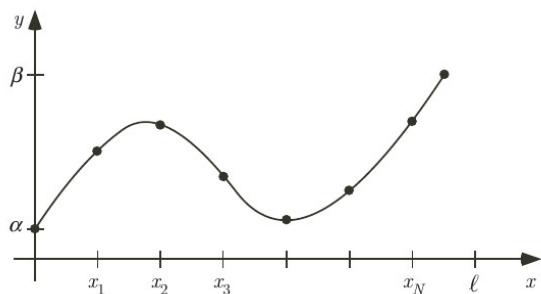
Chapter III: Solving systems of linear equations
_ Direct methods: (Cramer, Gauss, LU, ...)
_ Iterative methods (Jacobi, Gauss-Sidel...)



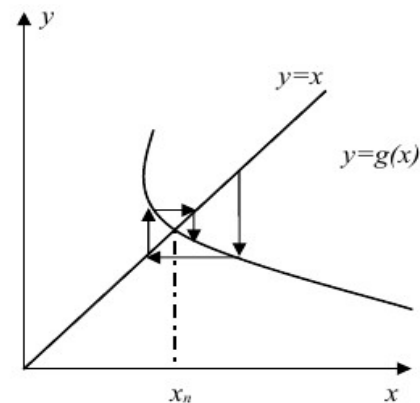
ROOT-ZERO-FINDING

A Transcendental Equation

Nonlinear Equations



(a)



(b)

Introduction : Problematic

Let $f : I = [a, b] \subseteq \mathbb{R} \rightarrow \mathbb{R}$, find $\alpha \in \mathbb{R}$ $f(\alpha) = 0$

Definition (A Transcendental Equation) :

For a polynomial $P(x)$, any real or complex argument x_0 such that $P(x_0) = 0$ is called a **root** of the polynomial. For a general real-valued function $f(x)$, it is customary to use different terminology: an x_0 such that $f(x_0) = 0$ is called a **zero** of the function.

Example : We want to determine the volume V occupied by a gas whose temperature is T and whose pressure is p . The equation of state (i.e. the equation linking p , V and T) is given by

$$[P + a(n/V)^2](V - Nb) = kNT$$

Introduction



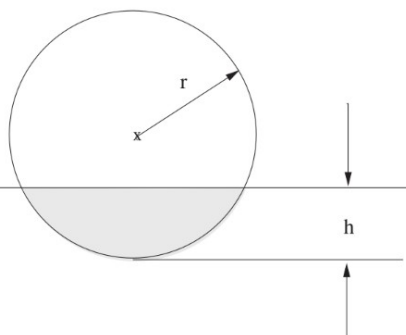
Historical Problem

Find square root of a number (Yale Babylonian 1700 BC) $\sqrt{\alpha}$

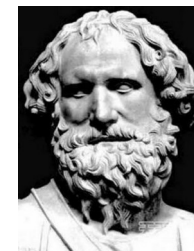


Archimedes Problem

$$h^3 - 3r h^2 + 4 \frac{\rho_s}{\rho_{water}} r^3 = 0$$

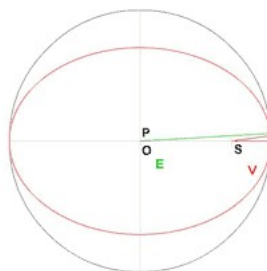


Archimedes
287 BC-212 BC



Kepler non linear equation

$$E - e \sin(E) - M = 0$$



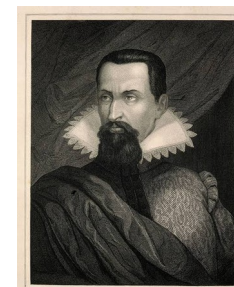
$V = 7.92$

$E = 3.33$

$M = 1$

$e = 0.7$

Kepler 1571-1630



Introduction

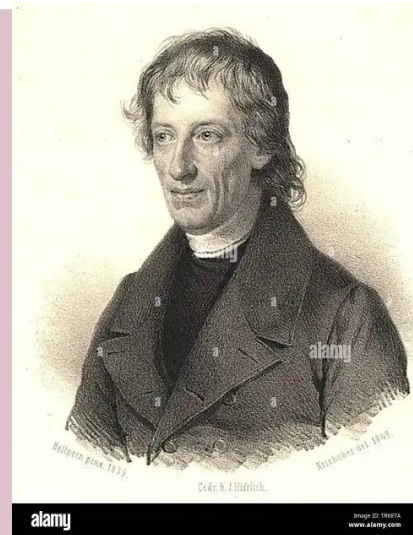
Find Solution : The approximate search for the solution of an equation is generally done in 3 steps

1 Existence Solution : which consists in establishing intervals $[a, b]$ as tightly as possible, each containing at least one zero.

Bolzano's Theorem:

If a continuous function $f(x)$ takes values of opposite signs at the endpoints of the interval $[a, b]$, that is, if $f(a) \cdot f(b) < 0$, then the interval contains **at least one zero** of the equation $f(x) = 0$.

(Intermediate Value Theorem)

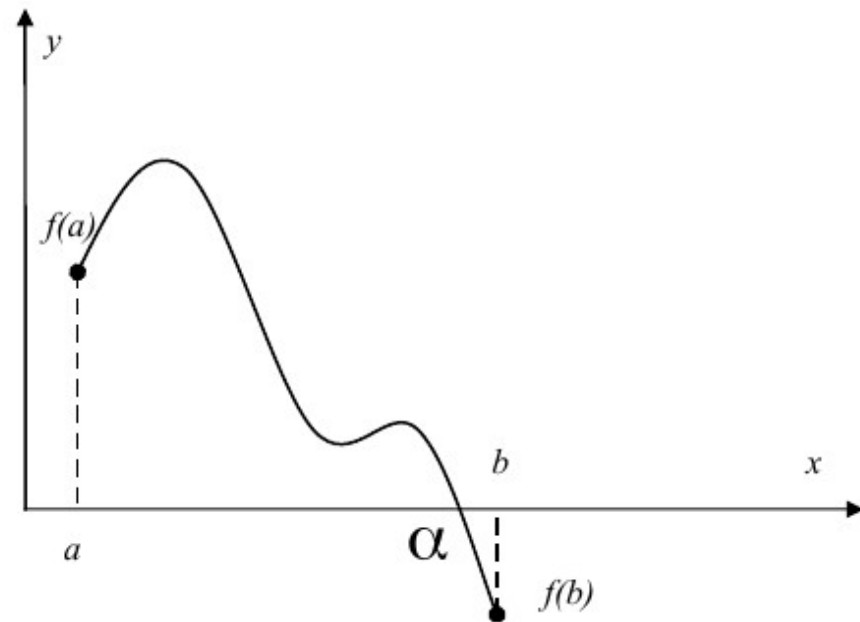


(Bernard Bolzano 1781-1848)

Introduction

2 Solution separation: which consists in establishing intervals $[a, b]$ as tightly as possible, each containing **one and only one root**.

3 Improving the accuracy or fine-tuning the approximate roots, i.e. obtaining their imposed accuracy..



Introduction

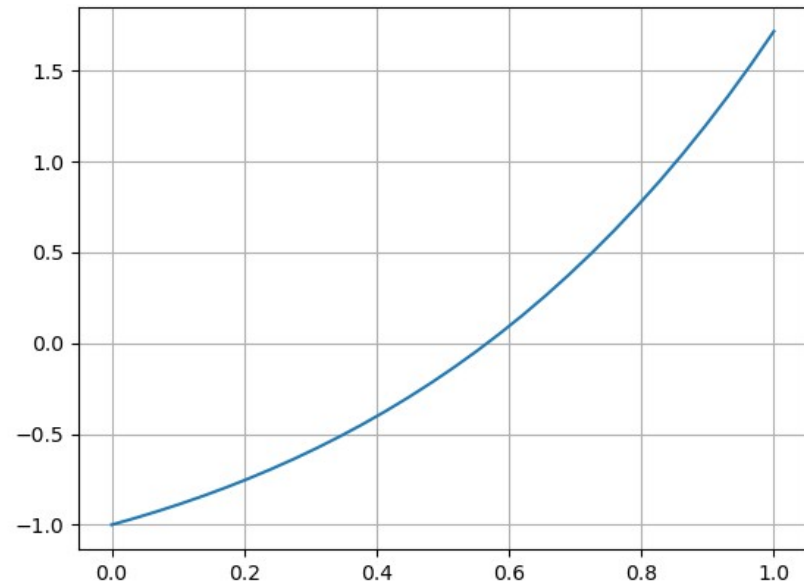
Uniqueness: A zero is unique provided that the function is continuous and monotonic on the interval $[a, b]$

Example: Investigate the existence and uniqueness of real zero of the following function in $[0,1]$

$$f(x) = xe^x - 1$$

$$f'(x) = (x+1)e^x$$

$$x \in [0.0, 1.0] \quad f'(x) > 0$$



Introduction

A value α is called a root of order n of the equation if it satisfies:

$$f(x) = (x - \alpha)^n g(x) \quad g(\alpha) \neq 0$$

Theorem 1 : If α is a root of order n of $f(x)$, then:

$$f(\alpha) = f'(\alpha) = f''(\alpha) = \dots = f^{[n-1]}(\alpha) = 0$$

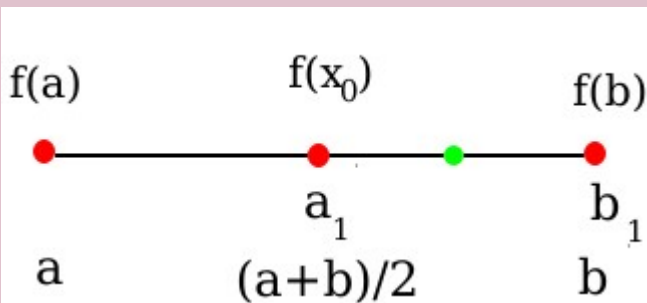
Theorem 2 : If $f \in C^1(I)$ has a root of order 2 $\alpha \in [a, b]$ and if f is continuous and differentiable on $[a, b]$ then : then α is a simple root of g :

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

Bisection Method : Dichotomie

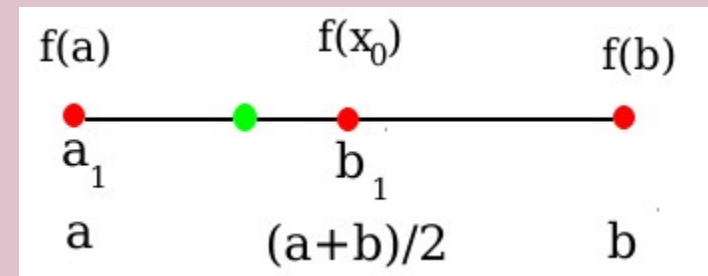
Description of the method:

We divide this interval into two part $[a_0, x_0]$ and $[x_0, b_0]$ where



$$x_0 = \frac{(a+b)}{2}$$

$$f(a) * f(x_0) < 0$$



if $f(x_0)=0$ then the solution is x_0 else , we seek the solution in the other two intervals by applying [Bolzano's Theorem](#) :

if $f(a_0).f(x_0)<0$ the needed solution is in this interval $[a_0, x_0]$ Denote $[a_1, b_1]$ where $a_1 = a_0$ et $b_1 = x_0$.

else $f(b_0).f(x_0)<0$ the solution is in the other interval $[x_0, b_0]$ named $[a_1, b_1]$ where $a_1 = x_0$ and $b_1 = b_0$ and we continue the search process until we obtain an interval $[a_n, b_n]$.

$$|b_n - a_n| < \epsilon$$

$$x_n = (a_n + b_n) / 2$$

Bisection Method

Iterations Numbers:

If the required precision is: ϵ then the number of iterations **N** in the bisection method is

$$N \geq \log_2 \left(\frac{|a-b|}{\epsilon} \right)$$

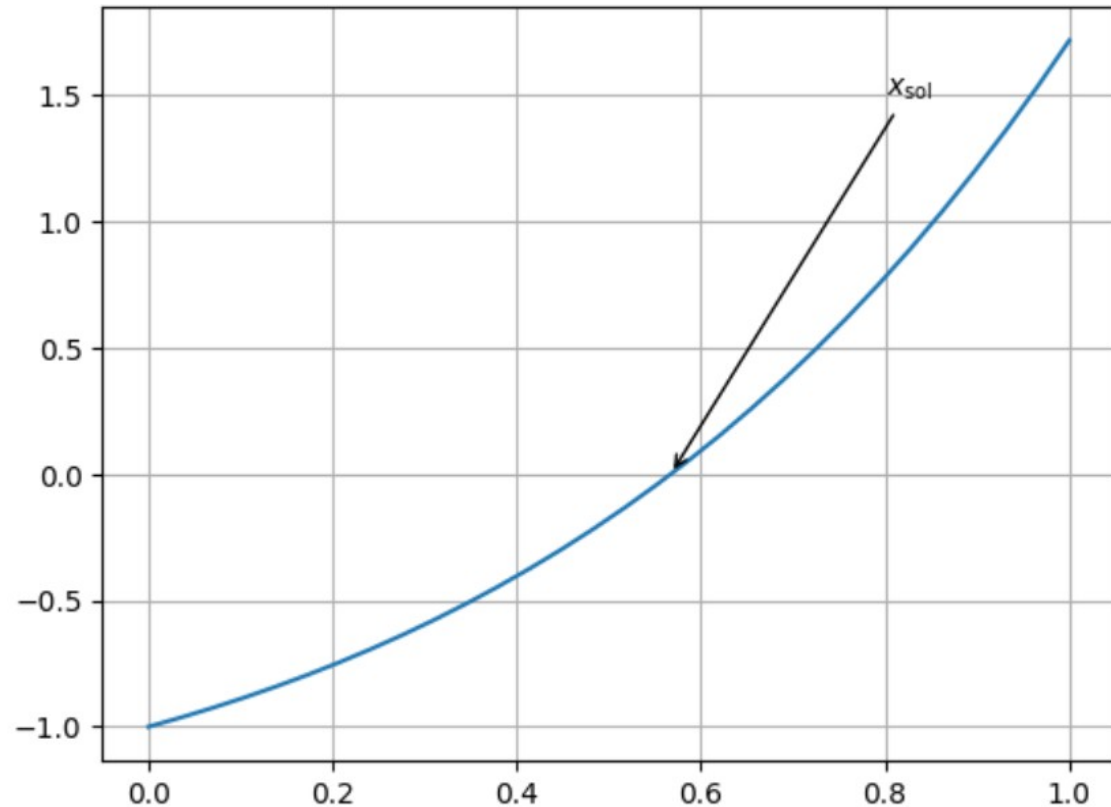
Bisection Method



Example: Determine the zero of $f(x)$ by the method of bisection for a 4 digit for :

$$f(x) = xe^x - 1$$

$$x \in [0.0, 1.0]$$



Bisection Method



i	a	x	b	f(a)	f(x)	f(b)
0	0.000000	0.500000	1.000000	-1.000000	-0.175639	1.718282
1	0.500000	0.750000	1.000000	-0.175639	0.587750	1.718282
2	0.500000	0.625000	0.750000	-0.175639	0.167654	0.587750
3	0.500000	0.562500	0.625000	-0.175639	-0.012782	0.167654
4	0.562500	0.593750	0.625000	-0.012782	0.075142	0.167654
5	0.562500	0.578125	0.593750	-0.012782	0.030619	0.075142
6	0.562500	0.570312	0.578125	-0.012782	0.008780	0.030619
7	0.562500	0.566406	0.570312	-0.012782	-0.002035	0.008780
8	0.566406	0.568359	0.570312	-0.002035	0.003364	0.008780
9	0.566406	0.567383	0.568359	-0.002035	0.000662	0.003364
10	0.566406	0.566895	0.567383	-0.002035	-0.000687	0.000662
11	0.566895	0.567139	0.567383	-0.000687	-0.000013	0.000662
12	0.567139	0.567261	0.567383	-0.000013	0.000325	0.000662
13	0.567139	0.567200	0.567261	-0.000013	0.000156	0.000325
13.28771237954945						

Bisection Method



Algorithm:

```
Begin
define  $a, b, f(x), \epsilon$ 
Do while  $|a-b| > \epsilon$ 
 $x = (a+b)/2$ 
If  $f(x) = 0$  process stop
End if
If  $f(a) \cdot f(x) < 0$        $b = x$ 
Else                     $a = x$ 
End if
End do
Write  $x$ 
```

Bisection Method



Python Code:

```
from math import *
def f(x): return x*exp(x)-1.0
a=0.0
b=1.0
tol=1.0E-4
i=0
while abs (b-a)>tol:
    x1 = (a+b)/2.0
    if f(a)*f(x1)<0:
        b=x1
    else:
        a=x1
    print (f"{i:2d}", " ", f"{a:2.8f}", " ", f"{b:2.8f}", " ", f"{x1:2.8f}", " ", f"{f(x1):2.8f}")
    i=i+1
print ("num iti=",i)
print ("x=",x1)
```



Regula-Falsi Methode

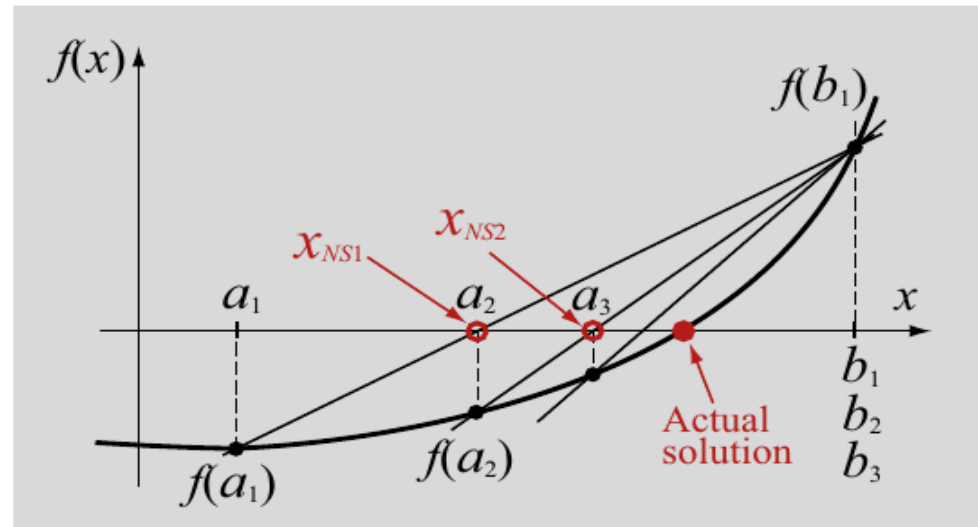
Regula Falsi:

In this method the search for the solution of $f(x)=0$ in the interval $[a, b]$ goes through the same process as the bisection method but instead of dividing the segment $[a,b]$ into two equal intervals, it makes more sense to divide it into the ratio $-f(a)/f(b)$ we thus obtain the value approximated to the zero:

$$y = \frac{f(b) - f(a)}{b - a}(x - b) + f(b)$$

$$x = a - f(a) \frac{(b - a)}{f(b) - f(a)}$$

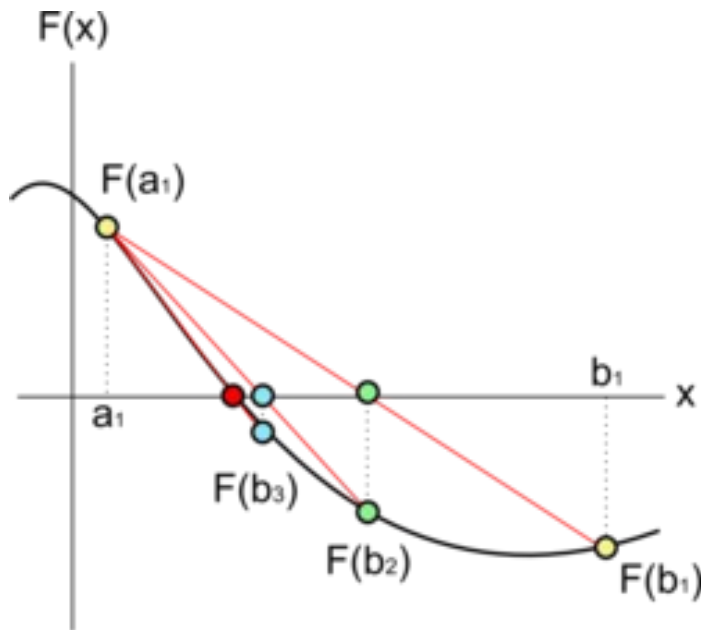
$$x_{NS} = \frac{af(b) - bf(a)}{f(b) - f(a)}$$



Regula Falsi Method



Algorithm



```
Begin
read a, b,
do
 $x = (af(b) - bf(a)) / (f(b) - f(a))$ 
if  $f(x) = 0$  stop processing
endif
if  $f(a) \cdot f(x) < 0$        $b = x$ 
else                     $a = x$ 
endif
while  $|f(x)| > e$ 
```

Secant's Method

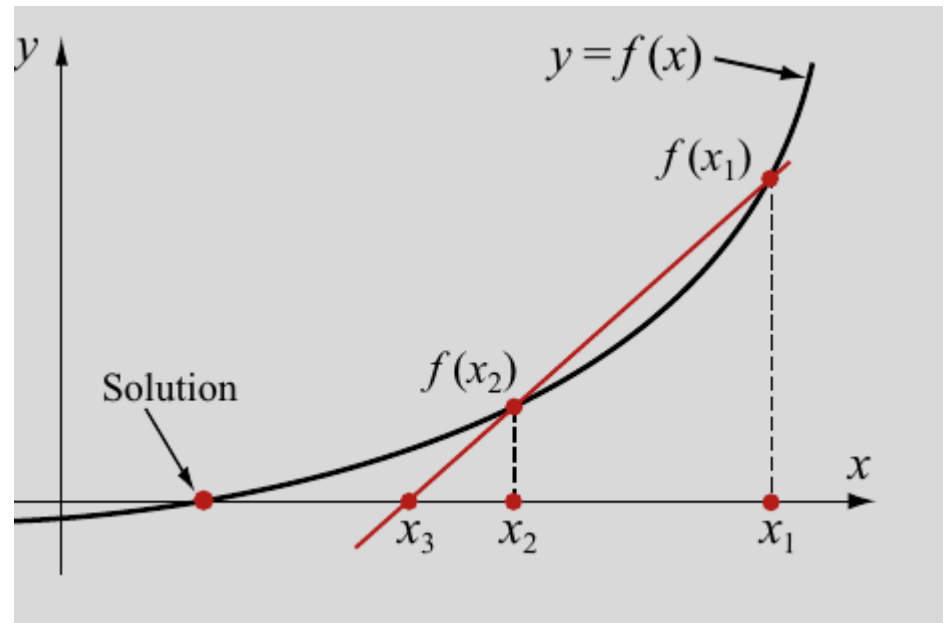


$$x = a - f(a) \frac{(b-a)}{f(b) - f(a)}$$

For a given $x_0, x_1 \in [a, b]$

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

$$|x_{i+1} - x_i| < \epsilon$$



Order of Convergence



Order of Convergence: A root-finding method providing a sequence

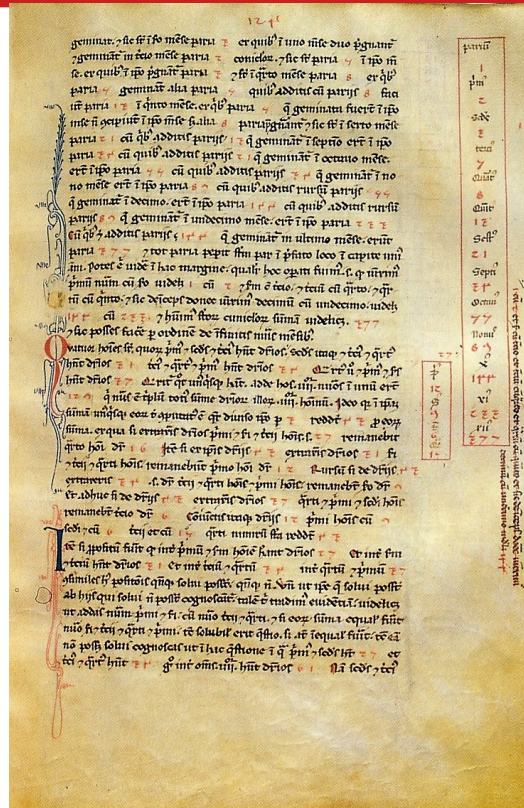
$\{x_k\}$ with $\lim_{k \rightarrow \infty} x_k = \alpha$ has order of convergence $p \geq 1$ if

$$\lim_{k \rightarrow \infty} \frac{|x_{k+1} - \alpha|}{|x_k - \alpha|^p} = C$$

For some positive bounded constant C , usually as the asymptotic error constant. In particular, if $p=1$ C must be smaller than unity for the criterion to be applicable



Leonardo Fibonacci
1170- 1250

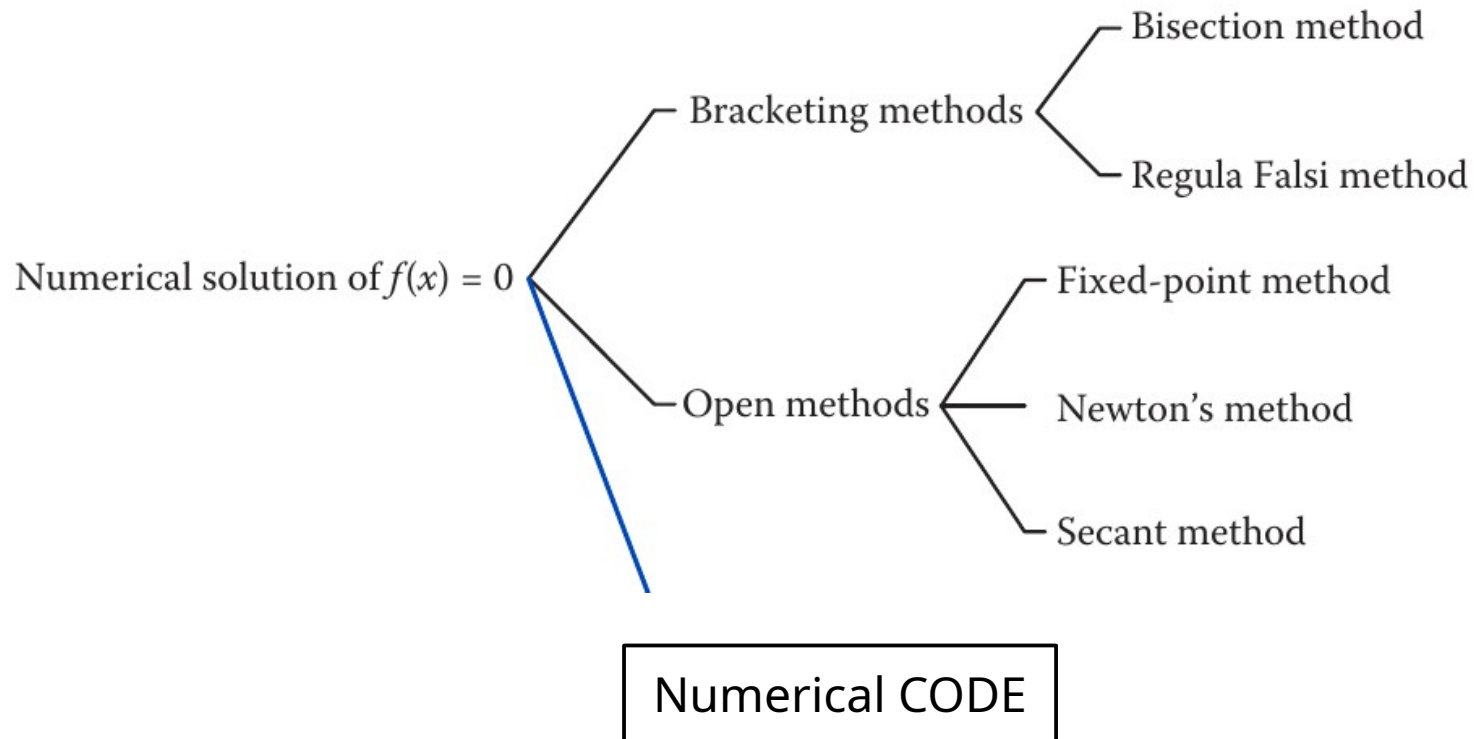


Liber Abaci,
secundum modum algebre et almuchabale



Dixit Algorizmi

Newton-Raphson's Method



Newton-Raphson's Method



Newton Method

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \frac{(x - x_0)^2}{2!}f''(x_0) + \frac{(x - x_0)^3}{3!}f^{[3]}(x_0) \dots$$

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \frac{(x - x_0)^2}{2!}f''(\xi)$$

$$f(x) = f(x_0) + (x - x_0)f'(x_0) \quad f(x) = 0$$

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

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

Newton-Raphson's Method



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

$$x_0 = \text{given}$$
$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Termination Criteria and errors estimation

$$|x_{k+1} - x_k| < \epsilon$$

$$\left| \frac{x_{k+1} - x_k}{x_k} \right| < \epsilon$$

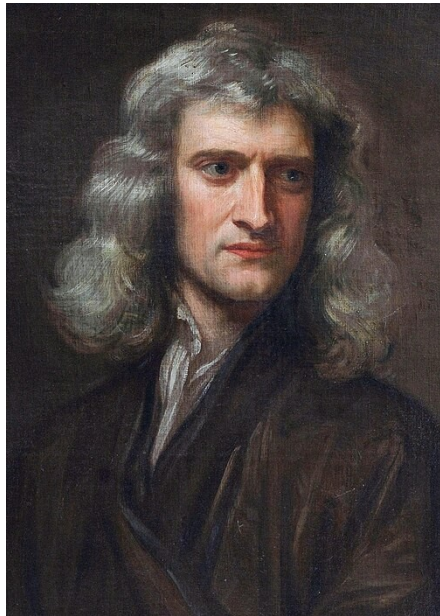
$$|f(x_k)| < \epsilon$$

Newton-Raphson's Method

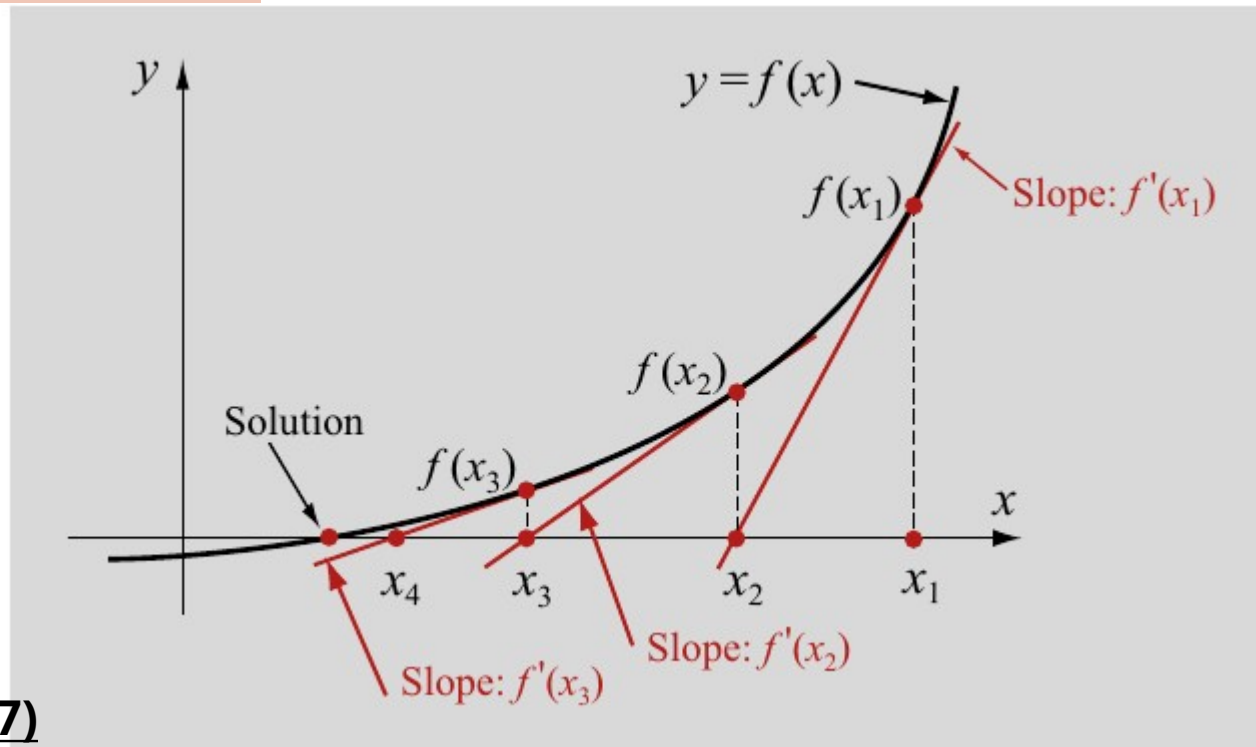


x_0 given

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



Isaac Newton (1642-1727)





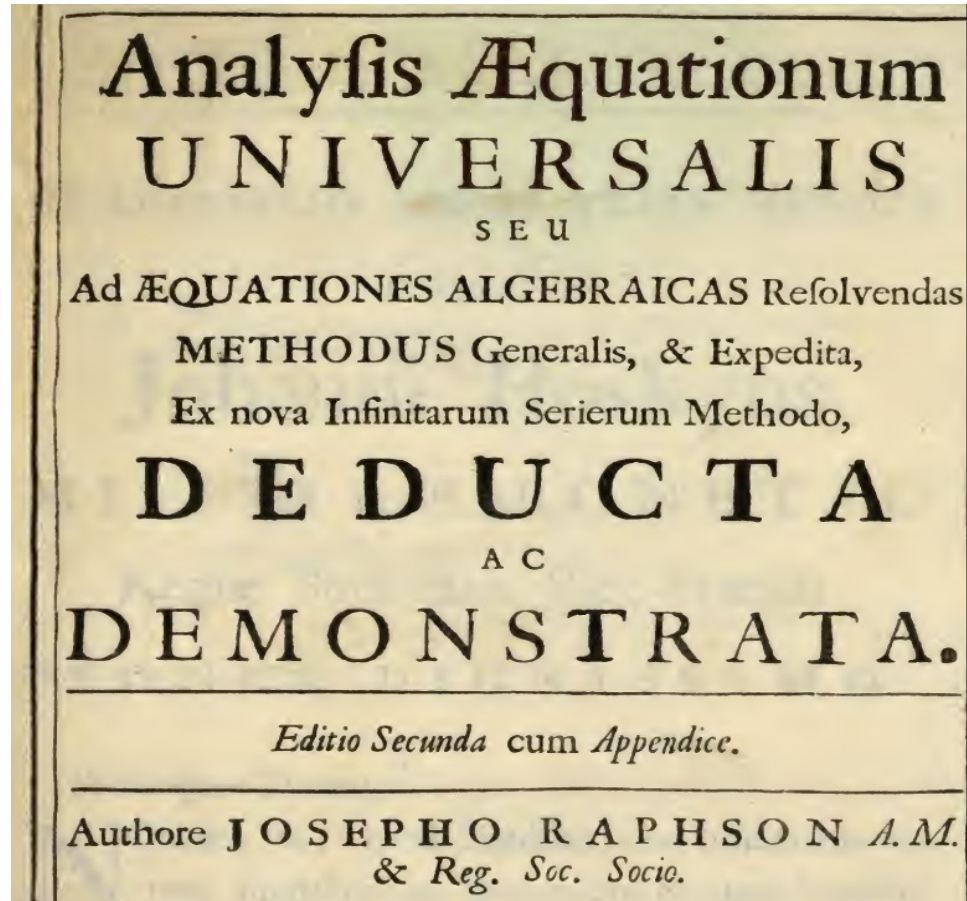
THE
METHOD of FLUXIONS
AND
INFINITE SERIES;
WITH ITS
Application to the Geometry of CURVE-LINES.

By the INVENTOR
Sir ISAAC NEWTON, *K^t*.
Late President of the Royal Society.

Newton-Raphson's Method



Josepho Raphson (1668-1715)



Newton-Raphson's Method



Algorithm

```
Begin
Def f,f'
read x0,
k=0
do
 $x_{k+1} = x_k - f(x_k)/f'(x_k)$ 
k=k+1
while  $|(x_k - x_{k-1})| > e$ 
```

Newton-Raphson's Method



Python Code:



```
from math import *
import numpy as np
def f(x): return x*np.exp(x)-1
def df(x): return (1.0+x)*np.exp(x)
x = np.zeros(100, float)
tol=1E-4
x[0]=0.0
k=0
for i in range(1,30,1):
    x[i]=x[i-1]-f(x[i-1]) /df(x[i-1])
    print (f"{i:2d}", " ",f"{x[i-1]:2.8f}", " ",f"{f(x[i-1]):2.8f}")
    if np.abs(x[i]-x[i-1]) < tol:
        k=i
        break
print ("iti=",k,"x=",x[k])
```

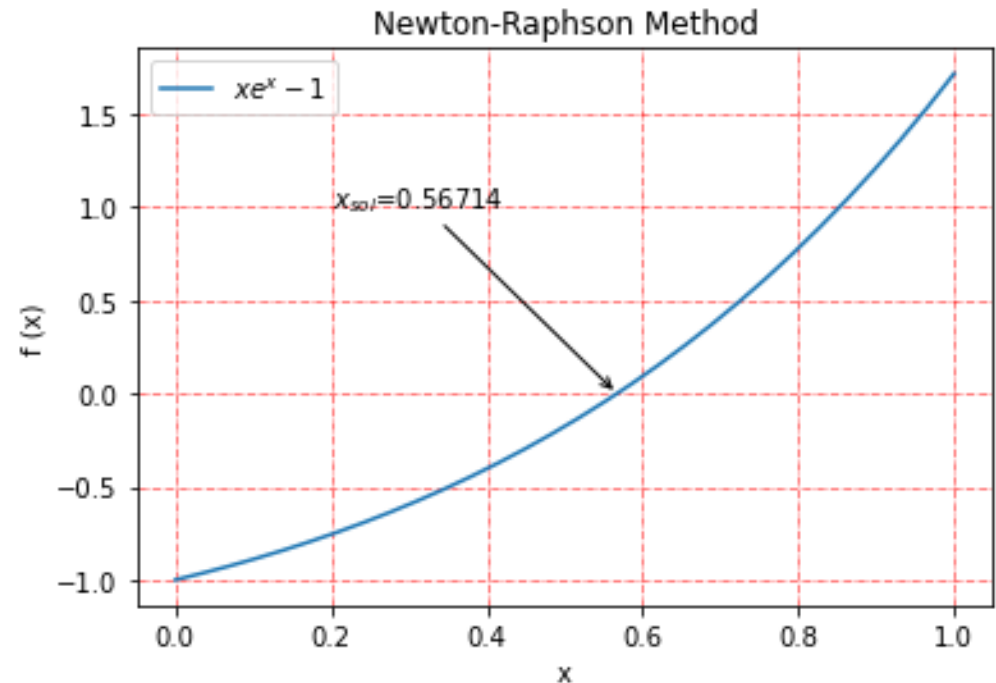
Newton-Raphson's Method



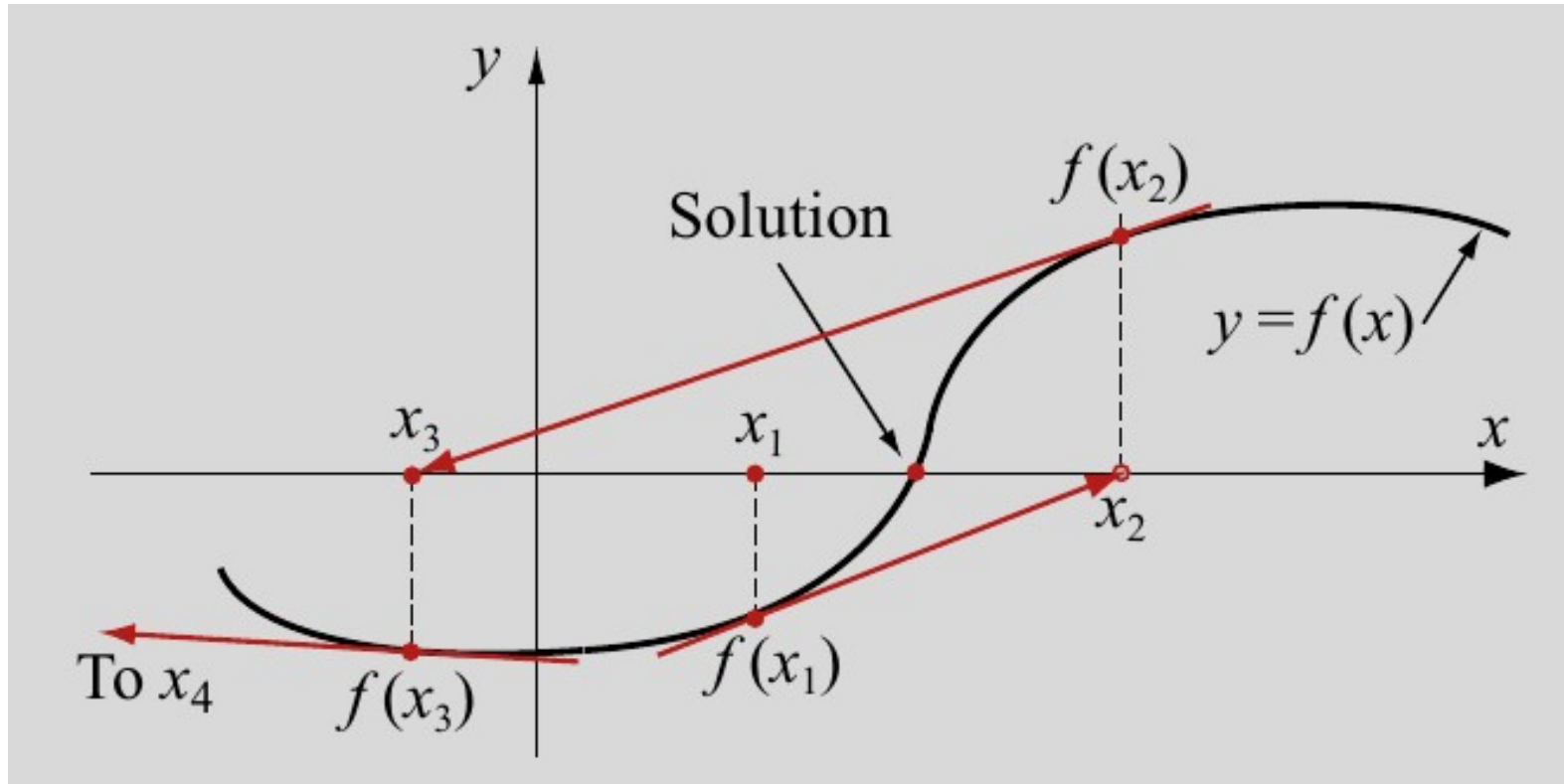
Python Code:

I	x(i)	f(x(i))
1	0.00000000	-1.00000000
2	1.00000000	1.71828183
3	0.68393972	0.35534255
4	0.57745448	0.02873389
5	0.56722974	0.00023889

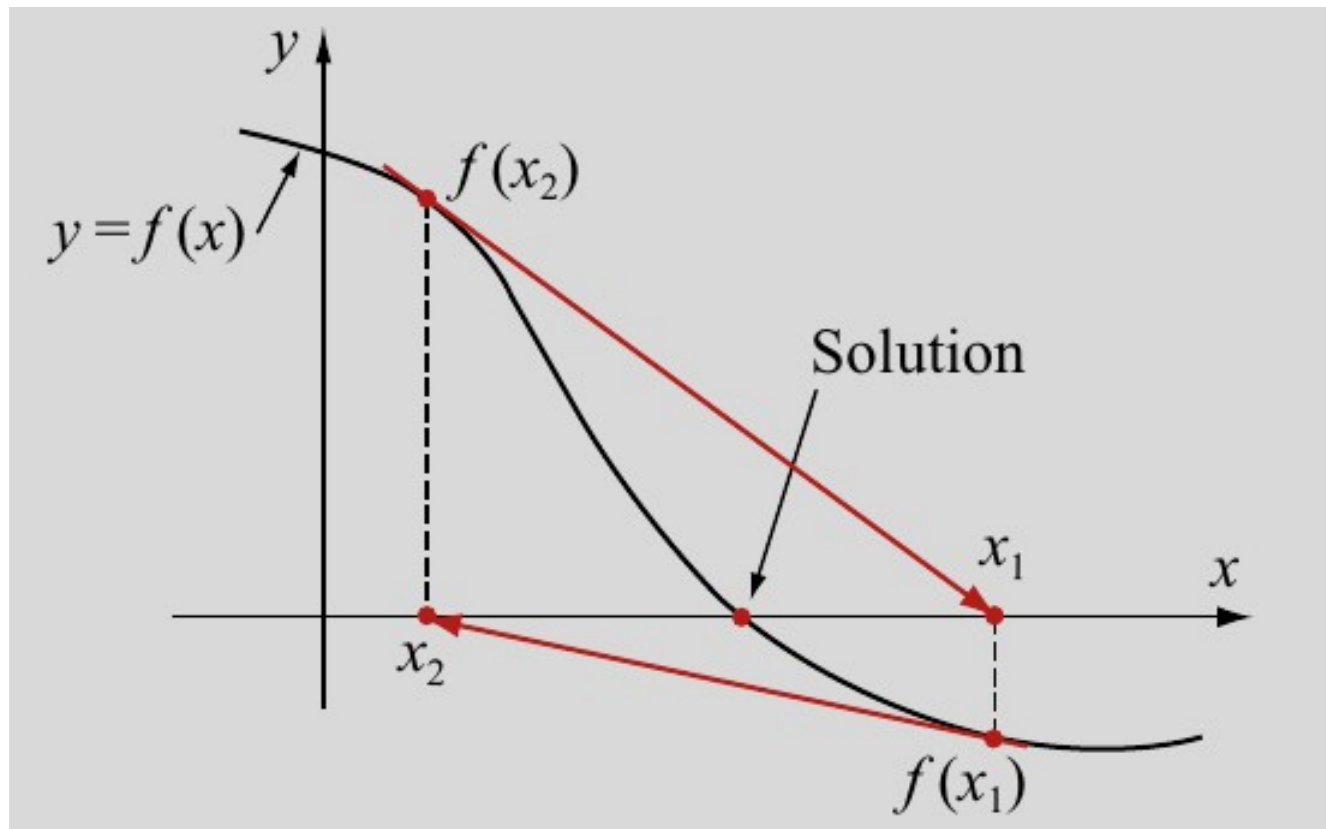
iti= 5 x= 0.5671432965302959



Newton-Raphson's Method



Newton-Raphson's Method



Newton-Raphson's Method



Order of Convergence: A root-finding method providing a sequence $\{x_k\}$ with $\lim_{k \rightarrow \infty} x_k = \alpha$ has order of convergence $p \geq 1$ if

$$\lim_{k \rightarrow \infty} \frac{|x_{k+1} - \alpha|}{|x_k - \alpha|^p} = C, \quad (1.11)$$

for some positive *bounded* constant C , usually termed as the *asymptotic error constant*. In particular, if $p = 1$, then C must be *smaller than unity* for the criterion to be applicable.

Newton-Raphson's Method



$$f(x) = f(x_n) + f'(x_n)(x - x_n) + \frac{1}{2}f''(\eta)(x - x_n)^2$$

$$0 = f(\alpha) = f(x_n) + f'(x_n)(\alpha - x_n) + \frac{1}{2}f''(\eta)(\alpha - x_n)^2$$

$$\alpha - \left(x_n - \frac{f(x_n)}{f'(x_n)}\right) = \frac{-f''(\eta)}{2f'(x_n)}(\alpha - x_n)^2 \qquad x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

$$x_{n+1} - \alpha = \frac{f''(\eta)}{2f'(x_n)}(x_n - \alpha)^2$$

$$C = \left| \frac{f''(\eta)}{2f'(x_n)} \right|$$

$$\lim_{n \rightarrow \infty} \frac{|x_{n+1} - \alpha|}{|(x_n - \alpha)^2|} = C$$

Newton-Raphson's Method



- **Newton's Local Convergence** conditions:

1. $f(a)f(b) < 0$: Existence of a root within I in the vicinity of x_0 .
2. $f \in C^2[a, b]$: Derivative continuity, the function $f(x)$ and its first derivative $f'(x)$ should be continuous within the interval I and the neighborhood of the root.
3. $\forall x \in I, f'(x) \neq 0$: The derivative does not vanish within an interval I containing the root.
4. $\forall x \in I, f''(x) \neq 0$: The derivative $f''(x)$ maintains a constant sign within the interval I .
5. $f(x_0)f''(x_0) > 0$

Newton-Raphson's Method



- **Newton's Global Convergence** conditions ($f(x_0)f''(x_0) < 0$):
 1. Conditions 1, 2, 3, and 4 of Newton's Local Convergence
 5. $\left| \frac{f(c)}{f'(c)} \right| \leq |b - a|, \begin{cases} c = a & |f'(a)| \leq |f'(b)| \\ c = b & |f'(a)| > |f'(b)| \end{cases}$
- **Kantorovich theorem** (convergence conditions for Newton's method):
 1. $f \in C^2[a, b]$
 2. $f'(x_0) \neq 0$
 3. $\max_{x \in [a, b]} |f''(x)| \left| \frac{f(x_0)}{f'(x_0)^2} \right| < 1/2$

Newton-Raphson's Method



3.5.3 Modified Newton's Method for Roots with Multiplicity 2 or Higher

If r is a root of $f(x)$ and r has a multiplicity 2 or higher, then convergence of the sequence generated by Newton's method is linear; see Theorem 3.2. In these situations, Newton's method may be modified to improve the speed of convergence. The modified Newton's method designed for roots of multiplicity 2 or higher is described as

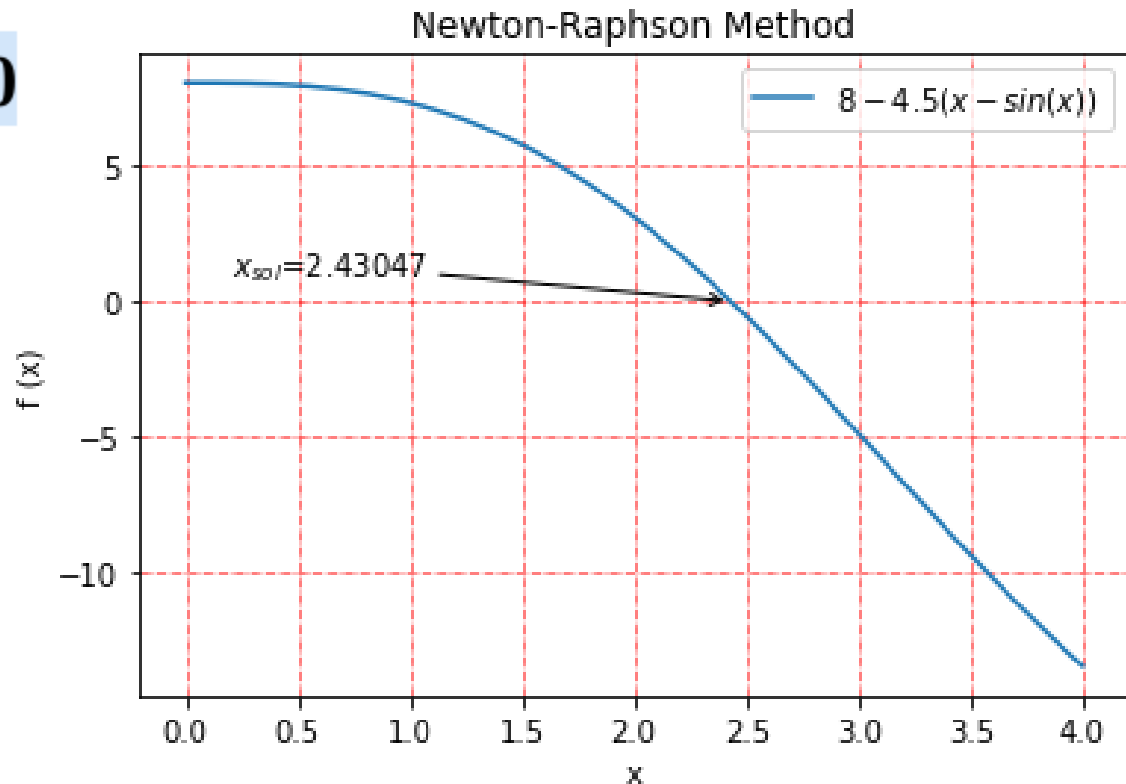
$$x_{n+1} = x_n - \frac{f(x_n)f'(x_n)}{[f'(x_n)]^2 - f(x_n)f''(x_n)}, \quad n = 1, 2, 3, \dots, \quad x_1 = \text{initial point}$$

Newton-Raphson's Method



Find by Newton Method the solution of the following problem:

$$8 - 4.5(x - \sin(x)) = 0$$



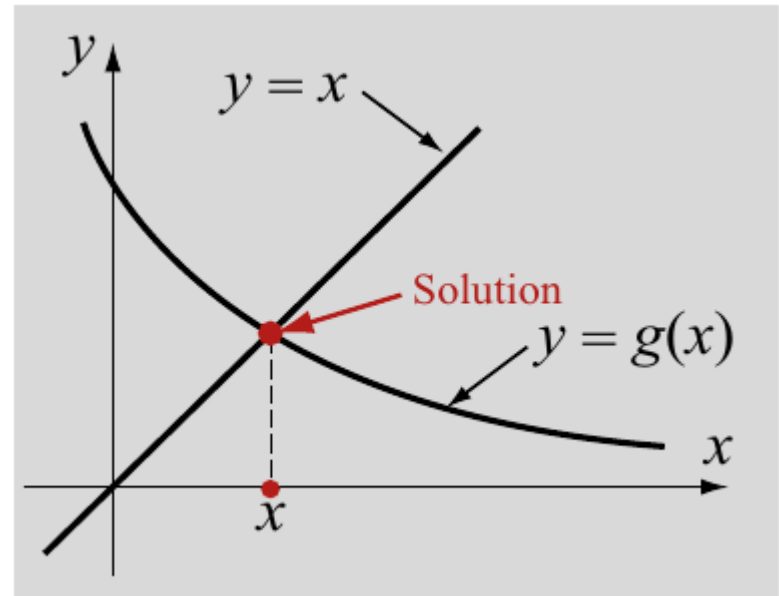
The fixed-point iteration method



Algorithm fixed-point iteration method.

$$f(x) = 0 \longrightarrow x = g(x)$$

$$\begin{array}{c} x_0 \\ x_{i+1} = g(x_i) \end{array}$$



$$|x_{i+1} - x_i| \leq \varepsilon_1 \quad \text{and/or} \quad |f(x_{i+1})| \leq \varepsilon_2$$

The fixed-point iteration method

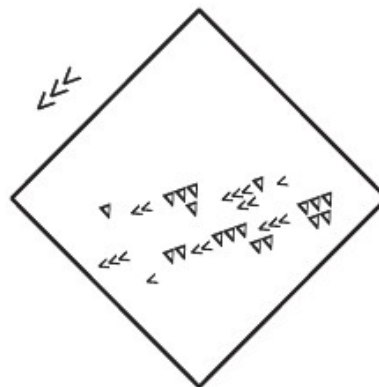


$$g(x) = \frac{1}{2} \left(\frac{a}{2} + x \right)$$

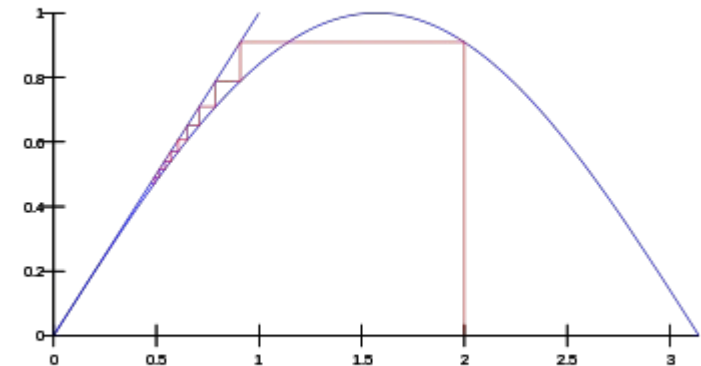
$$x_1 = \frac{1 + \frac{2}{1}}{2} = \frac{3}{2}.$$



(a)



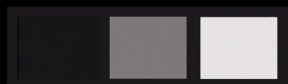
(b)



The fixed-point iteration method



YBC7289 - Obverse

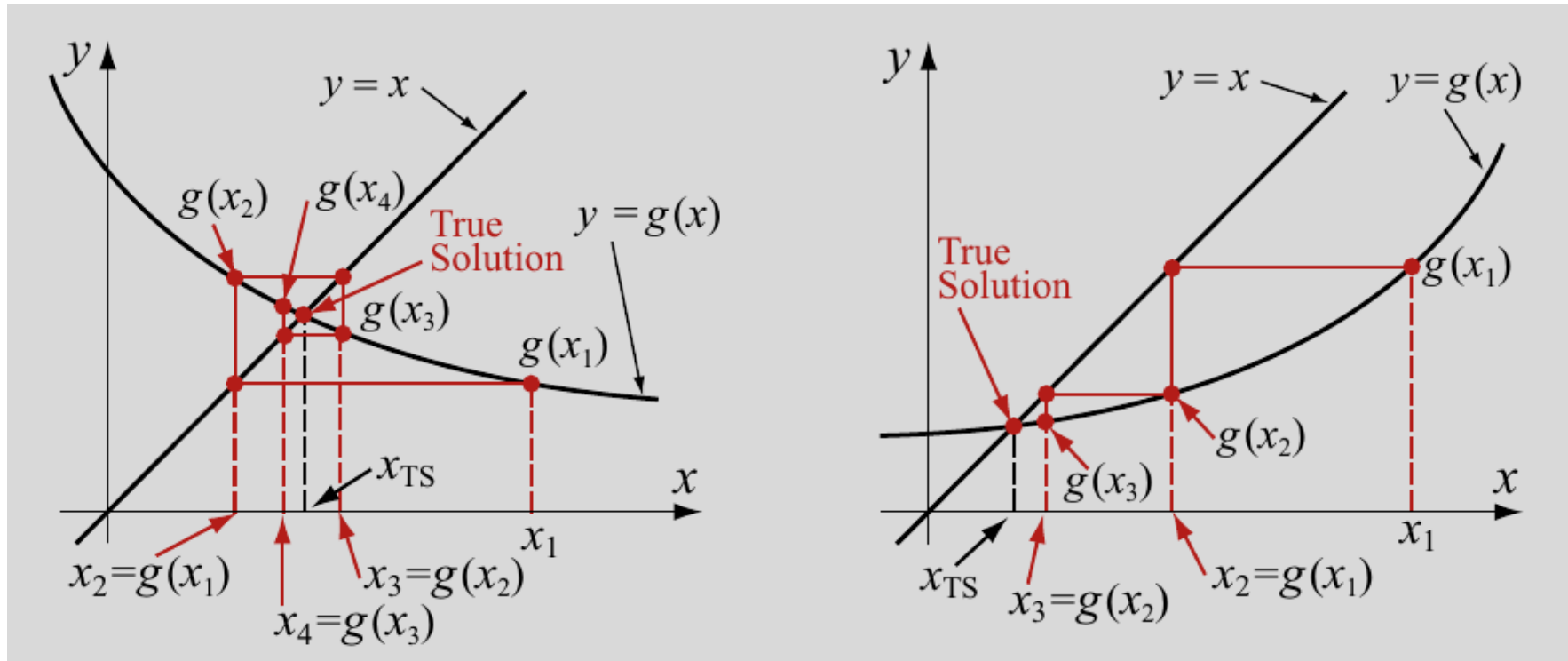


300 DPI



The fixed-point iteration method

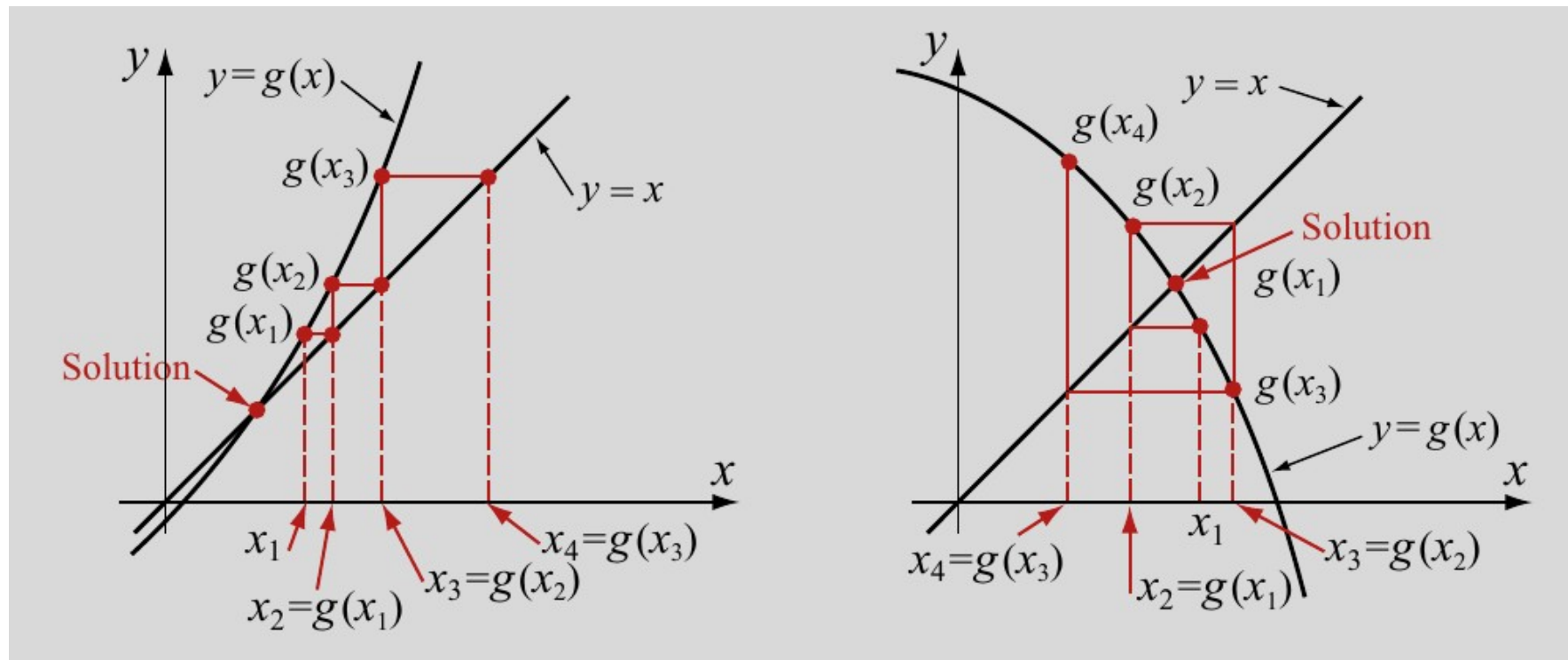
Convergence of the fixed-point iteration method.



The fixed-point iteration method



Divergence of the fixed-point iteration method.



The fixed-point iteration method



convergence of the fixed-point iteration method.

Definition 15.10 (contraction). Suppose that $-\infty < a < b < \infty$ and $g \in C([a, b])$. We say that g is **Lipschitz continuous**¹ on $[a, b]$ if and only if there exists a constant $L > 0$ such that

$$|g(x) - g(y)| \leq L|x - y|, \quad \forall x, y \in [a, b],$$

and the associated L is called the **Lipschitz constant**. The function g is called a **contraction** on $[a, b]$ if and only if it is Lipschitz on $[a, b]$ and its associated Lipschitz constant, L , satisfies $L \in (0, 1)$.

The fixed-point iteration method



Theorem 15.12 (uniqueness). *Suppose that $-\infty < a < b < \infty$, $g \in C([a, b])$, and $g([a, b]) \subseteq [a, b]$. If g is a contraction on $[a, b]$, then g has a unique fixed point $\xi \in [a, b]$. Furthermore, the sequence $\{x_k\}_{k=0}^{\infty}$ generated by (15.2) converges to ξ for any starting value $x_0 \in [a, b]$.*

Proof. Theorem 15.9 guarantees the existence of at least one fixed point $\xi \in [a, b]$. Suppose that $\eta \in [a, b]$ is another fixed point. Since g is a contraction,

$$|\xi - \eta| = |g(\xi) - g(\eta)| \leq L|\xi - \eta|.$$

Therefore,

$$0 \leq (1 - L)|\xi - \eta| \leq 0,$$

which proves that $\xi = \eta$. Hence, the fixed point $\xi \in [a, b]$ is unique.

The fixed-point iteration method



The fixed-point iteration method converges if, in the neighborhood of the fixed point, the derivative of $g(x)$ has an absolute value that is smaller than 1 (also called Lipschitz continuous):

$$|g'(x)| < 1 \quad (3.30)$$

The fixed-point iteration method



- **Convergence** to the root α occurs when the following conditions are satisfied:

$f(x) = 0 \iff x = g(x)$, equivalence criterion/condition

$\forall x \in [a, b], g(x) \in [a, b]$, domain containment condition

$\forall x \in [a, b], |g'(x)| \leq K < 1$, contraction mapping/condition

- **Divergence** occurs when:

$$\forall x \in [a, b], \min_{x \in [a, b]} |g'(x)| > 1$$

The fixed-point iteration method



- **Convergence is indefinite or uncertain:** in other cases where the above conditions are not met, the behavior of the fixed-point iteration method can be unpredictable, and convergence or divergence is not guaranteed.
- To determine **the number of iterations** required to achieve a desired precision ϵ , we use the formula:

$$|x_n - r| \leq \frac{K^N}{1 - K} |x_1 - x_0| \leq \epsilon, \quad K = \max_{x \in [a, b]} |g'(x)|$$

$$N \geq \frac{\ln \left[\frac{1 - k}{|x_1 - x_0|} \epsilon \right]}{\ln k}$$

The fixed-point iteration method

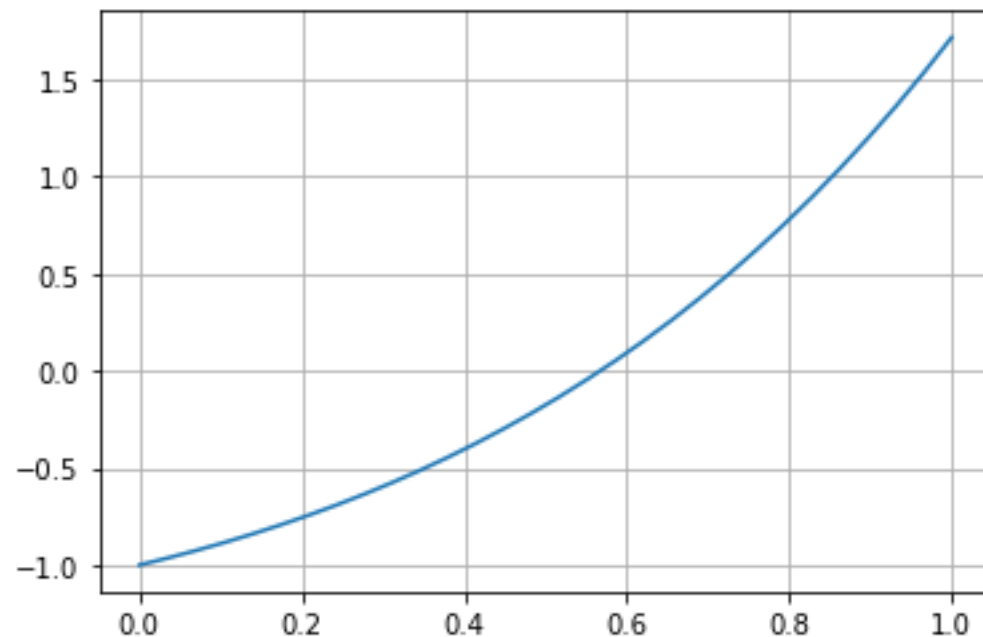


$$f(x) = xe^x - 1$$

I	x	g(x)
1	1.00000000	1.71828183
2	0.36787944	-0.46853639
3	0.69220063	0.38309147
4	0.50047350	-0.17446790
5	0.60624354	0.11156623
6	0.54539579	-0.05903351
.....		
14	0.56690891	-0.00064752
15	0.56727623	0.00036739
16	0.56706790	-0.00020831
17	0.56718605	0.00011816
18	0.56711904	-0.00006701

$$X_0 = 1.0$$

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

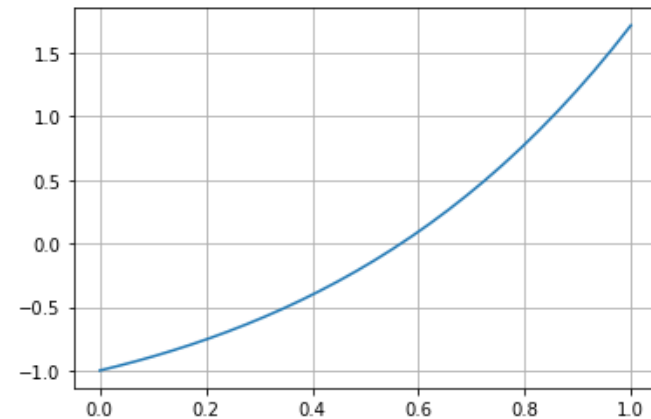


The fixed-point iteration method



```
from math import *
import numpy as np
import matplotlib.pyplot as plt
def f(x): return x*np.exp(x)-1
def g(x): return np.exp(-x)
x = np.zeros(100, float)
tol=1E-4
x[0]=0.0
k=0
for i in range(1,30,1):
    x[i]=g(x[i-1])
    print (f"{i:2d}", " ", f"{x[i]:2.8f}", " ", f"{f(x[i]):2.8f}")
    if np.abs(x[i]-x[i-1]) < tol:
        k=i
        break
print ("iti=",k,"x=",x[k])
```

$$x_0 = 1.0$$
$$x_{i+1} = \exp(-x_i)$$

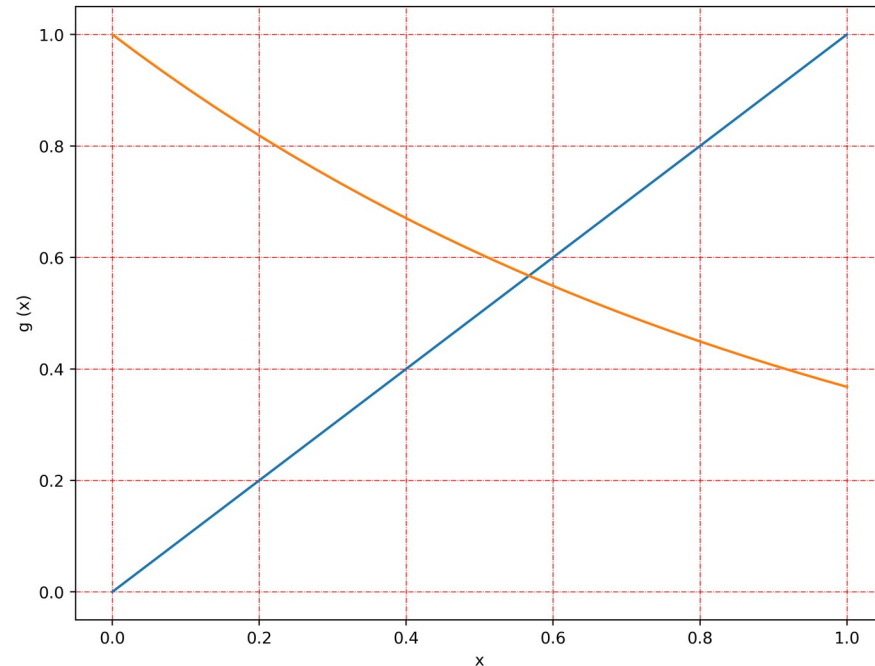


$$f(x) = xe^x - 1$$

The fixed-point iteration method



```
#to draw the curve
x = np.linspace(0, 1, 30)
y = np.exp(-x)
plt.figure(figsize=(9,7), dpi=500)
plt.ylabel('f (x)')
plt.xlabel('x')
plt.plot(x,x,x,y)
plt.grid('on',color='r', linestyle='-.',linewidth=0.5)
plt.show()
```



$$f(x) = xe^x - 1$$

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