The convergence of the Regula Falsi Method

Name: Abhirup Bag
Department: CSE-B
Roll number: 13000122082
Subject: Numerical Methods
Subject Code: OEC-IT601A

CONTENT

Introduction

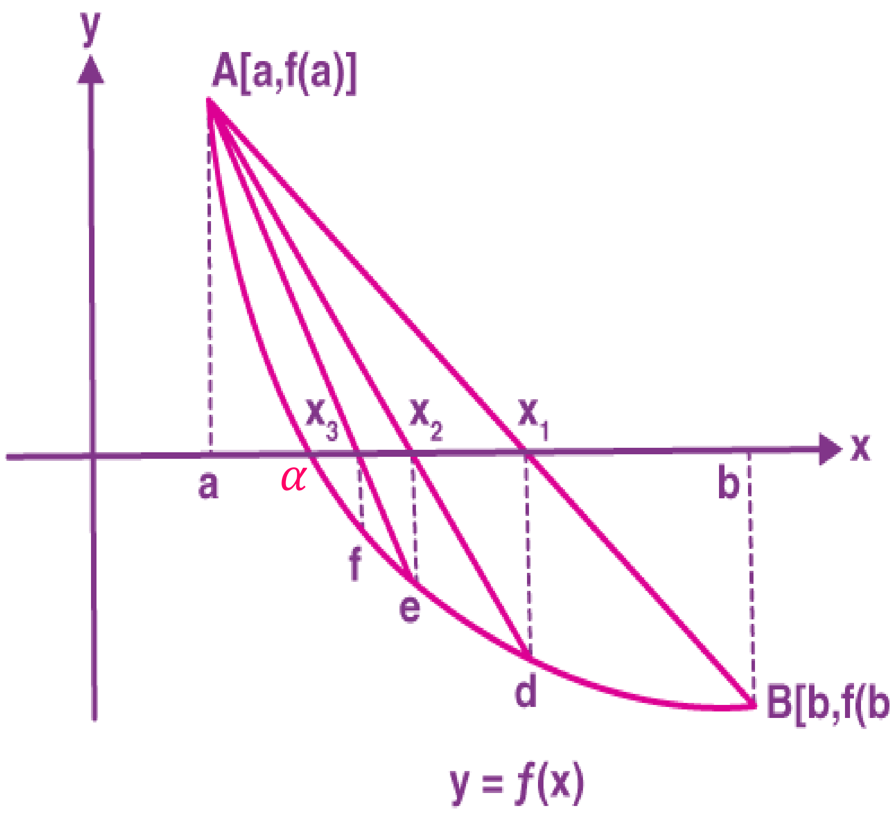
The Regula Falsi Method, also known as the False Position Method, is a numerical approach used to determine the roots of a non-linear equation, represented as $f(x) = 0$.

This method involves selecting two initial estimates, a_0 and b_0, such that the function values at these points have opposite signs, which suggests the presence of a root within that interval. It operates on the premise that if a continuous function crosses zero within an interval, there must be a root located somewhere in that span. Compared to the bisection method, this method offers quicker convergence, but it is not as fast as the Newton-Raphson Method.

The Regula Falsi Method is particularly effective for continuous functions when a root is situated between two specific points.

Derivation/Formulation and geometrical interpretation

Let us consider an equation f(x)=0 with graphical representation as,



Now, let α be a real root of the equation y = f(x) = 0, so clearly f(α) = 0.

To find the real root of the equation f(x) = 0, we consider a sufficiently small interval (a, b) where a < b such that f(a) and f(b) will have opposite signs. According to the intermediate value theorem, this implies a root lies between a and b.

Also, the curve y = f(x) will meet the x-axis at a certain point between A[a, f(a)] and B[b, f(b)].

Now, the equation of the chord joining A[a, f(a)] and B[b, f(b)] is given by:

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

Let y = 0 be the point of intersection of the chord equation (given above) with the x-axis. Then,

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

This can be simplified as:

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

$$\Rightarrow \left(\frac{af(a) - bf(a)}{f(b) - f(a)}\right) + a = x$$

$$\Rightarrow x = \frac{af(a) - bf(a) + af(b) - af(a)}{f(b) - f(a)}$$

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

Thus, the first approximation is x_1 = [a f(b) – b f(a)] / [f(b) – f(a)]

Also, x_1 is the root of f(x) if f(x_1) = 0.

If f(x_1) ≠ 0 and if f(x_1) and f(a) have opposite signs, then we can write the second approximation as:

x_2 = [a f(x_1) – x_1 f(a)] / [f(x_1) – f(a)]

Similarly, we can estimate x_3, x_4, x_5, and so on.

Convergence Analysis

- **Oder of Convergence of Iterative Process :-**

Let α be the real root of the equation $f(x) = 0$ and e_i be the small quantity by which x_i differs from α. Then,

$$x_i - \alpha = e_i \text{ (e_i is called error at the } i^{th} \text{ approximation)}$$

$$\text{So, } x_{i+1} - \alpha = e_{i+1}$$

The order of convergence of an iterative process is p, if p is the largest number such that

$$\lim_{i\to\infty}\left(\frac{e_{i+1}}{(ei)^p}\right) \leq k, \text{ where k is a finite number.}$$

- **Order of Convergence of Regula Falsi Method :-**

Let α be the real root of the equation $f(x) = 0$ and e_{i-1} be the small quantity by which x_{i-1} differs from α. Then,

$$\left.\begin{array}{c} x_{i-1} - \alpha = e_{i-1} \\ x_i - \alpha = e_i \\ x_{i+1} - \alpha = e_{i+1} \end{array}\right\}$$

Putting the values of x_{i-1}, x_i and x_{i+1} in Regula Falsi iterative formula i.e.,

$$x_{i-1} = \frac{x_{i-1}f(x_i) - x_i f(x_{i-1})}{f(x_i) - f(x_{i-1})}, \text{ we get,}$$

$$e_{i+1} + \alpha = \frac{(e_{i-1} + \alpha) f(e_i + \alpha) - (e_i + \alpha) f(e_{i-1} + \alpha)}{f(e_i + \alpha) - f(e_{i-1} + \alpha)}$$

$$= \frac{e_{i-1} \,.\, f(e_i + \alpha) \; - e_i \,.\, (e_{i-1} + \alpha) + \alpha f(e_i + \alpha) - \alpha f(e_{i-1} + \alpha)}{f(e_i + \alpha) - f(e_{i-1} + \alpha)}$$

$$= \frac{e_{i-1} \,.\, f(e_i + \alpha) \; - e_i \,.\, f(e_{i-1} + \alpha)}{f(e_i + \alpha) - f(e_{i-1} + \alpha)} + \alpha$$

$$\Rightarrow \qquad e_{i+1} = \frac{e_{i-1} \,.\, f(e_i + \alpha) \; - e_i \,.\, f(e_{i-1} + \alpha)}{f(e_i + \alpha) - f(e_{i-1} + \alpha)} \qquad \ldots\ldots\ldots (2)$$

Now, expanding $f(e_i + \alpha)$ and $f(e_{i-1} + \alpha)$ by Taylor's Theorem, the numerator of (2) i.e.,

$$e_{i-1} \,.\, f(e_i + \alpha) \; - e_i \,.\, f(e_{i-1} + \alpha)$$

$$= \; e_{i-1} \left[f(\alpha) + e_i f'(\alpha) + \frac{e_i^2}{2!} f''(\alpha) + \cdots \right] - e_i \left[f(\alpha) + e_{i-1} f'(\alpha) + \frac{e_{i-1}^2}{2!} f''(\alpha) + \; \cdots \right]$$

[Now, putting f(α) = 0 and neglecting all terms greater than the degree of 2 as e_i and e_{i-1} are very small]

$$= \frac{e_{i-1}\, e_i^2}{2!} f''(\alpha) - \frac{e_i\, e_{i-1}^2}{2!} f''(\alpha)$$

$$= \frac{e_i\, e_{i-1}\,(e_i - e_{i-1})}{2} f''(\alpha) \qquad \cdots\cdots\cdots (3)$$

Again, the denominator of (2) i.e.,

$$f(e_i + \alpha) - f(e_{i-1} + \alpha) = \left[f(\alpha) + e_i f'(\alpha) + \frac{e_i^2}{2!} f''(\alpha) + \cdots \right] - \left[f(\alpha) + + e_{i-1} f'(\alpha) \frac{e_{i-1}^2}{2!} f''(\alpha) + \cdots \right]$$

$$= (e_i - e_{i-1})\, f'(\alpha) \qquad \cdots\cdots\cdots (4)$$

[Terms containing e_i^2 , e_{i-1}^2 and higher degree terms are neglected]

Using (3) and (4), equation (2) becomes,

$$e_{i+1} = \frac{e_i\, e_{i-1}\,(e_i - e_{i-1})}{2\,(e_i - e_{i-1})} \cdot \frac{f''(\alpha)}{f'(\alpha)}$$

Or,

$$e_{i+1} = \frac{e_i \; . e_{i-1}}{2} \cdot \frac{f''(\alpha)}{f'(\alpha)} = c_{i-1} \cdot e_i \cdot k \qquad \cdots\cdots\cdots (5)$$

Where $k = \frac{f''(\alpha)}{2f'(\alpha)}$ is a finite constant.

Let p be the order of convergence, then $\frac{e_{i-1}}{e_i^p} = k'$, where k' is finite.

Or,

$$e_{i+1} = k' e_i^{\mathrm{p}} \qquad \cdots\cdots\cdots (6)$$

And,

$$e_i = k' e_{i-1}^{\mathrm{p}}$$

Or,

$$e_{i-1} = \left[\frac{e_i}{k'}\right]^{\frac{1}{p}} \qquad \cdots\cdots\cdots (7)$$

Putting the values of e_{i+1} and e_{i-1} in (5), we get

$$k' e_i^{\mathrm{p}} = \left[\frac{e_i}{k'}\right]^{\frac{1}{p}} \cdot e_i \cdot k$$

$$\Rightarrow \quad k' e_i^{\mathrm{p}} = \frac{k}{(\,k'\,)^{(1/p)}} \cdot e_i^{\,(1+\frac{1}{p})} \qquad \cdots\cdots\cdots (8)$$

Choosing k and k' such that, $k' = \frac{k}{k'^{(1/p)}}$, equation (8) becomes

$$e_i^p = e_i^{\left(1+\frac{1}{p}\right)}$$

$$\therefore p = 1 + \frac{1}{p}$$

$$\Rightarrow p^2 - p - 1 = 0$$

$$\therefore p = \frac{1 \pm \sqrt{1+4}}{2}$$

Taking positive sign, we have $p = \frac{1+\sqrt{5}}{2} = 1 \cdot 618$

So, Order of convergence of Regula Falsi Method is 1·618.

Key Points :-

- ✓ The method is at least linearly convergent, meaning the error reduces proportionally in each iteration.
- ✓ In some cases, it can be super-linearly convergent (faster than linear but slower than quadratic).
- ✓ However, in cases where one side of the interval remains fixed (i.e., if the function is nearly flat on one side), the convergence can become extremely slow.

Slow Convergence Issue :-

- ✓ If one endpoint remains unchanged for multiple iterations, the newly computed points keep getting closer to the root but at a decreasing rate.
- ✓ This happens when the function is asymmetric or nearly tangent to the x-axis at one endpoint, causing Regula Falsi to behave inefficiently compared to methods like the secant method or Newton's method.

❑ Modifications to Improve Convergence

To address slow convergence, variations have been developed:

- **Illinois Algorithm**
  - ✓ If the same endpoint remains fixed, it reduces its weight by half.
  - ✓ This forces the interval to update more dynamically, accelerating convergence.

- **Anderson-Björck Method**
  - ✓ Adjusts the secant approximation to avoid stagnation.

- **Modifications Using Dynamic Weighting**
  - ✓ Introduce factors that adjust weights dynamically based on function values.

Error Analysis

The error in the Regula Falsi method follows a **linear convergence** pattern, meaning the error in each iteration is proportional to the error in the previous iteration:

$$E_{n+1} = CE_n$$

where C is a constant ($0<C<1$).

- This means that the error decreases at a **fixed rate** per iteration.
- The method is generally faster than **Bisection**, which also has linear convergence.
- However, it is **slower than Newton's method**, which has **quadratic convergence** $(E_{n+1} = CE_n^2)$

Application And Implementation

❖ Applications :-

The **Regula Falsi (False Position) method** is widely used in engineering, physics, finance, and other fields where root-finding is required. Some common applications include:

- ❑ **Engineering Applications**
  - **Electrical Engineering:** Used in circuit analysis to find operating points (e.g., solving nonlinear equations in transistor circuits).
  - **Mechanical Engineering:** Applied in stress-strain analysis, vibration analysis, and fluid mechanics problems.
  - **Civil Engineering:** Used in structural analysis and load distribution calculations.

- ❑ **Physics Applications**
  - **Quantum Mechanics:** Used in solving Schrödinger's equation for energy eigenvalues.
  - **Classical Mechanics:** Applied in solving projectile motion equations with drag forces.
  - **Thermodynamics:** Used in heat transfer calculations, such as solving radiation and conduction equations.

- **Mathematical and Computational Applications**
    - **Root-finding Problems:** Used when an equation cannot be solved algebraically.
    - **Numerical Analysis:** Serves as a benchmark for comparing root-finding methods.
- **Finance and Economics Interest**
    - **Rate Calculations:** Helps in determining the internal rate of return (IRR).
    - **Risk Analysis:** Used in models predicting market trends based on mathematical equations.

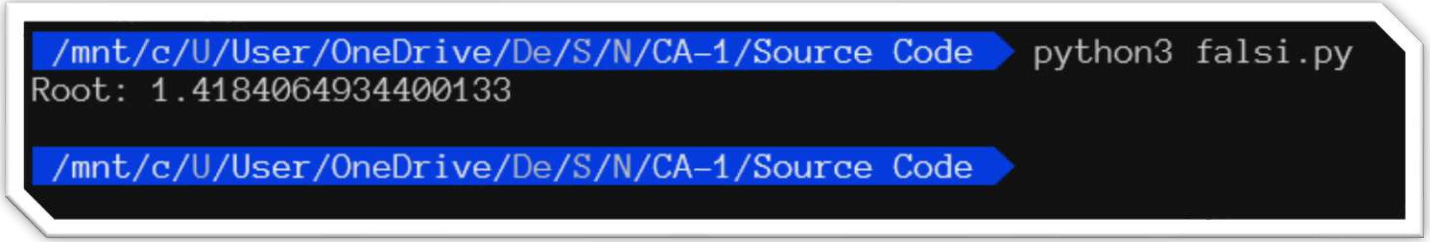
❖ Code Implementation In Python And Output :-

```python
import math
def regula_falsi(f, a, b, tol=1e-6, max_iter=100):
    if f(a) * f(b) >= 0:
        raise ValueError("Root is not bracketed. Choose different a and b.")
    for i in range(max_iter):
        c = b - (f(b) * (b - a)) / (f(b) - f(a))  # Secant formula
        if abs(f(c)) < tol:  # Check convergence
            return c
        if f(a) * f(c) < 0:  # Root lies between a and c
            b = c
        else:  # Root lies between c and b
            a = c
    return c  # Return last approximation

# Define the function
def f(x):
    return math.log10(x) – cos(x);

a = 1 # Initial guesses
b = 2

root = regula_falsi(f, a, b)
print("Root:", root)
```



❖ Worked Out Example :-

Let us consider an equation, f(x) = $\log_{10}^{x} - \cos x$. Now we need to compute a real root of this equation using Regula Falsi Method correct up to 3 decimal places.

x	1	2
f(x)	-0.5403	0.71718

So, f(1) f(2) < 0, hence Initial guess interval : $(a_0 , b_0) = (1 , 2)$

n	a_n (-ve)	b_n (+ve)	$f(a_n)$	$f(b_n)$	x_{n+1}	$f(x_{n+1})$
0	1	2	-0.54030	0.71718	1.42967	0.01458
1	1	1.42967	-0.54030	0.01458	1.41838	-0.00003
2	1.41838	1.42967	-0.00003	0.01458	1.41841	−0.000008
3	1.41840	1.42967	−0.000008	0.01458	1.41841	−0.000008

∴ Approximate root of f(x) = 0 using Regula Falsi method correct up to 3 decimal places is 1.418

References

- "**Numerical Methods for Scientists and Engineers**" – R. W. Hamming
- "**Applied Numerical Analysis**" – Curtis F. Gerald & Patrick O. Wheatley
- "**An Introduction to Numerical Analysis**" – Kendall E. Atkinson
- “**Analysis of Numerical Methods**” - Isaacson and Keller
- “**Numerical Mathematics and Computing**” by E. Ward Cheney and David R. Kincaid
- "**Numerical Analysis**" – Richard L. Burden & J. Douglas Faires
- "**Numerical Methods for Engineers**" – Steven C. Chapra & Raymond P. Canale

Thank You