Numerical Analysis Series

False Position Method

(Also known as Method of Regula Falsi)

Author
Shahaduddin
shahaduddin.com

Date
February 2, 2026
Version 1.0

1. Theoretical Background

Mathematical Definition

The **False Position Method** (or *Regula Falsi*) is the oldest method for finding the real root of an equation $f(x) = 0$.
Unlike the Bisection method, which divides the interval in half, this method uses a linear interpolation. We consider two points $(a, f(a))$ and $(b, f(b))$ such that $f(a)$ and $f(b)$ have opposite signs. The chord joining these points intersects the x-axis at a point which gives a closer approximation to the root.

Derivation of the Formula: The equation of the straight line (chord) passing through points $A(a, f(a))$ and $B(b, f(b))$ is:

$$\frac{y - f(a)}{x - a} = \frac{f(b) - f(a)}{b - a} \tag{1}$$

The point where this line cuts the x-axis corresponds to $y = 0$. Substituting $y = 0$ and solving for x gives the first approximation x_1:

$$x_1 = \frac{af(b) - bf(a)}{f(b) - f(a)} \tag{2}$$

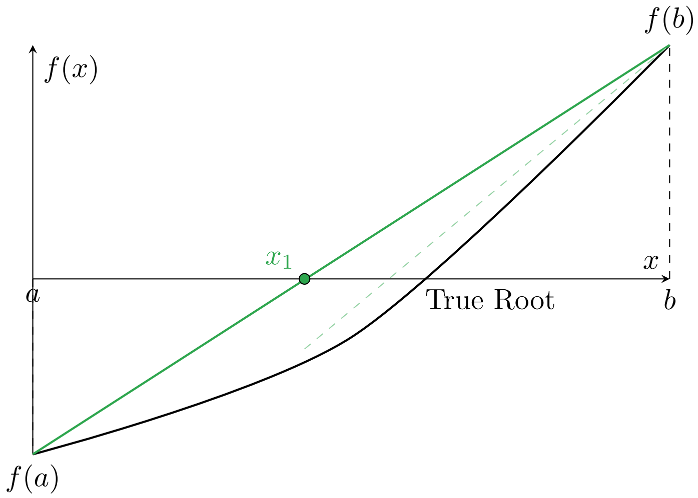

Fig 1. Geometric interpretation: The intersection of the chord determines x_1.

2. Manual Calculation Example

Problem: Find a real root of the equation $xe^x - 3 = 0$ correct to three decimal places using the False Position method.

Solution: Let $f(x) = xe^x - 3$. We first determine the interval by finding signs:

- $f(1) = 1 \cdot e^1 - 3 = 2.718 - 3 = -0.2817$ (< 0)
- $f(1.5) = 1.5 \cdot e^{1.5} - 3 = 6.722 - 3 = 3.7225$ (> 0)

The root lies between 1 and 1.5. So, $a = 1$ and $b = 1.5$.

Iteration 1: Using the formula $x_1 = \frac{af(b)-bf(a)}{f(b)-f(a)}$:

$$x_1 = \frac{1(3.7225) - 1.5(-0.2817)}{3.7225 - (-0.2817)} = \frac{3.7225 + 0.4225}{4.0042} = \frac{4.145}{4.0042} \approx \mathbf{1.035}$$

Now check the sign at x_1:

$$f(1.035) = 1.035e^{1.035} - 3 = -0.0864 \quad (-ve)$$

Since $f(1.035)$ is negative and $f(1.5)$ is positive, the root lies between 1.035 and 1.5.

Iteration 2: Set $a = 1.035$ and $b = 1.5$:

$$x_2 = \frac{1.035(3.7225) - 1.5(-0.0864)}{3.7225 - (-0.0864)} \approx \mathbf{1.045}$$

$$f(1.045) = -0.0286 \quad (-ve)$$

Iteration 3: New interval is $[1.045, 1.5]$:

$$x_3 = \frac{1.045(3.7225) - 1.5(-0.0286)}{3.7225 - (-0.0286)} \approx \mathbf{1.048}$$

*Proceeding in this manner, the root converges to approximately **1.049**.*

3. Code Implementations

Note: The following code demonstrates solving $f(x) = x^3 - x - 2 = 0$ in the range $[1, 2]$.

>PythonImplementation (.py)

```
def false_position(f, a, b, tol=1e-6, max_iter=100):
    """
    False Position (Regula Falsi) Method.
    Uses linear interpolation to find roots.
    """
    if f(a) * f(b) >= 0:
```

```python
        raise ValueError("Root not bracketed")

    for i in range(max_iter):
        # Linear interpolation formula
        c = (a * f(b) - b * f(a)) / (f(b) - f(a))

        if abs(f(c)) < tol:
            return c

        if f(c) * f(a) < 0:
            b = c
        else:
            a = c

    return c

# Example Usage
func = lambda x: x**3 - x - 2
root = false_position(func, 1, 2)
print(f"Root: {root}")
```

>FortranImplementation (.f90)

```fortran
program false_position
    implicit none
    real :: a, b, c, fc, tol
    integer :: i

    ! Initial bracket
    a = 1.0; b = 2.0; tol = 1e-6

    if (f(a) * f(b) >= 0) then
        print *, "Error: Root not bracketed."
        stop
    end if

    do i = 1, 100
        ! Interpolation formula
        c = (a * f(b) - b * f(a)) / (f(b) - f(a))
        fc = f(c)

        if (abs(fc) < tol) exit

        if (fc * f(a) < 0) then
            b = c
        else
            a = c
        end if
    end do

    print *, "Root found: ", c

contains
    real function f(x)
        real, intent(in) :: x
        f = x**3 - x - 2.0
    end function f
end program false_position
```

>C++Implementation (.cpp)

```cpp
#include <iostream>
#include <cmath>
#include <iomanip>
#include <functional>

/**
 * False Position Method (Regula Falsi) implementation.
 */
double false_position(std::function<double(double)> f, double a, double b,
   double tol = 1e-6) {
    if (f(a) * f(b) >= 0) {
        std::cerr << "Error: Root must be bracketed." << std::endl;
        return NAN;
    }

    double c;
    for (int i = 0; i < 100; ++i) {
```

```cpp
            // Linear interpolation formula
            c = (a * f(b) - b * f(a)) / (f(b) - f(a));

            if (std::abs(f(c)) < tol) return c;

            if (f(c) * f(a) < 0) b = c;
            else a = c;
        }
        return c;
}

int main() {
        auto f = [](double x) { return std::pow(x, 3) - x - 2; };
        double root = false_position(f, 1.0, 2.0);

        std::cout << std::fixed << std::setprecision(6);
        std::cout << "Root: " << root << std::endl;
        return 0;
}
```

For more numerical analysis resources, visit shahaduddin.com/PyNum