Numerical Analysis Series

Fixed-Point Iteration

(Also known as Functional Iteration)

Author
Shahaduddin
shahaduddin.com

Date
February 2, 2026
Version 1.0

1. Theoretical Background

Definition

A number p is a **fixed point** for a given function g if $g(p) = p$.
The connection to root-finding is simple: Given a problem $f(x) = 0$, we can rewrite it in the form $x = g(x)$. If we find a fixed point p for g, then p is also a zero of $f(x)$.
The Iterative Formula: To approximate the fixed point, we choose an initial approximation p_0 and generate the sequence $\{p_n\}_{n=0}^{\infty}$ by letting:

$$\boxed{p_n = g(p_{n-1}) \quad \text{for } n \geq 1} \tag{1}$$

Convergence Criteria

Not all choices of $g(x)$ will result in a converging sequence. According to the **Fixed-Point Theorem**:

- **Existence:** If $g \in C[a, b]$ and $g(x) \in [a, b]$ for all $x \in [a, b]$, then g has at least one fixed point in $[a, b]$.
- **Uniqueness & Convergence:** If, in addition, $g'(x)$ exists on (a, b) and there exists a constant k with $0 < k < 1$ such that:

  $$|g'(x)| \leq k < 1 \quad \text{for all } x \in (a, b)$$

  Then the fixed point is unique and the iteration converges for any p_0 in $[a, b]$.

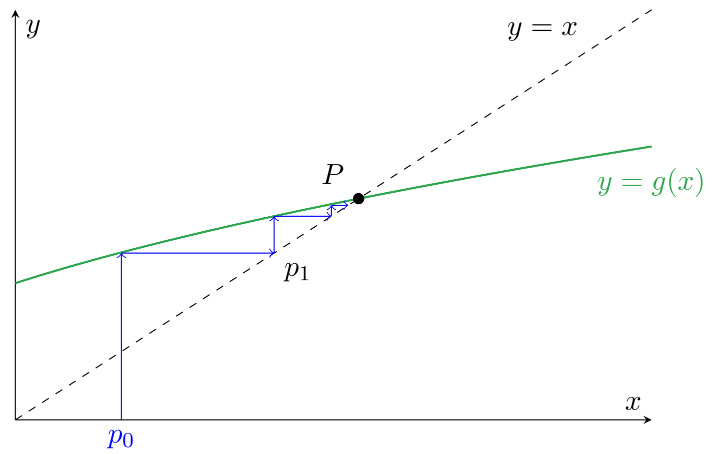

Fig 1. "Staircase" convergence where $0 < g'(x) < 1$.

2. Algorithm

Fixed-Point Iteration Algorithm

To find a solution to $p = g(p)$ given an initial approximation p_0:

1. **Input:** Initial approximation p_0; tolerance TOL; max iterations N_0.
2. **Step 1:** Set $i = 1$.
3. **Step 2:** While $i \leq N_0$ do Steps 3–6:
   - **Step 3:** Set $p = g(p_0)$ (Compute p_i).
   - **Step 4:** If $|p - p_0| < TOL$ then **OUTPUT** p; **STOP**.
   - **Step 5:** Set $i = i + 1$.
   - **Step 6:** Set $p_0 = p$ (Update).
4. **Step 7:** Output "Method failed after N_0 iterations". **STOP**.

3. Code Implementations

Example: Finding the Golden Ratio by solving $x^2 - x - 1 = 0$. Two rearrangements are used: $g(x) = 1 + 1/x$ (Python) and $g(x) = \sqrt{x+1}$ (Fortran/C++).

>PythonImplementation *(.py)*

```python
def fixed_point_iteration(g, x0, tol=1e-6, max_iter=100):
    """
    Solves x = g(x) given an initial guess x0.
    Find roots by rearranging f(x)=0 into x = g(x).
    """
    x = x0
    for i in range(max_iter):
        x_new = g(x)

        if abs(x_new - x) < tol:
            return x_new

        x = x_new

    print("Warning: Max iterations reached")
    return x

# Example: Root of x^2 - x - 1 = 0 -> x = 1 + 1/x
g = lambda x: 1 + 1/x
root = fixed_point_iteration(g, 1.5)
print(f"Root: {root}")
```

>FortranImplementation (.f90)

```fortran
program fixed_point
    implicit none
    real :: x, x_next, tol
    integer :: i, max_iter

    ! Parameters
    x = 1.0
    tol = 1e-6; max_iter = 100

    print *, "Solving x = g(x) via Fixed Point Iteration..."

    do i = 1, max_iter
        x_next = g(x)

        ! Convergence check
        if (abs(x_next - x) < tol) exit

        x = x_next

        if (i == max_iter) print *, "Warning: Max iterations reached."
    end do

    print *, "Root found: ", x_next

contains
    real function g(x)
        real, intent(in) :: x
        ! Example: Solving x^2 - x - 1 = 0 => x = sqrt(x + 1)
        ! This converges to the Golden Ratio (approx 1.618034)
        g = sqrt(x + 1.0)
    end function g
end program
```

>C++Implementation (.cpp)

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

/**
 * Fixed Point Iteration method for solving x = g(x).
 * Converges if |g'(x)| < 1 in the neighborhood of the root.
 */
double fixed_point_iteration(std::function<double(double)> g, double x0,
   double tol = 1e-6, int max_iter = 100) {
    double x = x0;
    for (int i = 0; i < max_iter; ++i) {
        double x_next = g(x);

        // Stop if the difference is within tolerance
        if (std::abs(x_next - x) < tol) {
            return x_next;
        }

```

```
        x = x_next;
    }
    std::cout << "Warning: Max iterations reached." << std::endl;
    return x;
}

int main() {
    // Example: Solving x^2 - x - 1 = 0 => x = sqrt(x + 1)
    // The fixed point of this g(x) is the Golden Ratio.
    auto g = [](double x) { return std::sqrt(x + 1.0); };

    double root = fixed_point_iteration(g, 1.0);

    std::cout << std::fixed << std::setprecision(6);
    std::cout << "Fixed Point Root: " << root << std::endl;

    return 0;
}
```

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