Numerical Analysis Series

Incremental Search Method

(Finding Initial Root Brackets)

Author
Shahad Uddin
shahaduddin.com

Date
February 2, 2026
Version 1.0

1. Theoretical Background

Definition

The **Incremental Search Method** (also known as the Tabulation Method) is a brute-force approach used to find intervals (brackets) that contain a root of the equation $f(x) = 0$.
It relies on the **Intermediate Value Theorem**: If a continuous function $f(x)$ changes sign between two points x_1 and x_2 (i.e., $f(x_1) \cdot f(x_2) < 0$), then there is at least one root in the interval $[x_1, x_2]$.

The Procedure:

1. Choose a starting point a and an ending point b.
2. Choose a step size Δx (or h).
3. Evaluate the function at successive steps: $x_{i+1} = x_i + \Delta x$.
4. Check if $f(x_i) \cdot f(x_{i+1}) \leq 0$. If true, a root exists in $[x_i, x_{i+1}]$.

Limitations:

- **Step Size Dilemma:** If Δx is too large, you might jump over two roots close to each other (missing them). If Δx is too small, the computation becomes inefficient.
- It provides a *range*, not a precise answer. It is usually used as a first step for methods like Bisection or Newton-Raphson.

2. Code Implementations

The code below scans the range $[0, 5]$ to solve $e^x - 3x = 0$.

>PythonImplementation *(.py)*

```python
import math

def incremental_search(f, a, b, step=0.1):
    """
    Detailed Incremental Search for Transcendental Equations.
    Scans [a, b] to find sub-intervals where the function changes sign.
    """
    brackets = []
    x1 = a
    f1 = f(a)

    while x1 < b:
        x2 = min(x1 + step, b)
        f2 = f(x2)

        # Check for root or sign change
        if f1 * f2 <= 0:
            brackets.append((x1, x2))

        x1 = x2
        f1 = f2

    return brackets

# Example: Solve e^x - 3x = 0
f = lambda x: math.exp(x) - 3*x
print(f"Brackets: {incremental_search(f, 0, 5, 0.2)}")
```

>FortranImplementation (*.f90*)

```fortran
program incremental_search
    implicit none
    real :: a, b, h, x1, x2, f1, f2

    ! Range and Step Size
    a = 0.0
    b = 5.0
    h = 0.2

    x1 = a
    f1 = f(x1)

    print *, "Searching for root brackets in [", a, ",", b, "]..."

    do while (x1 < b)
        x2 = min(x1 + h, b)
        f2 = f(x2)

        ! Intermediate Value Theorem check
        if (f1 * f2 <= 0.0) then
            print *, "Root bracket found: [", x1, ",", x2, "]"
        end if

        x1 = x2
        f1 = f2
    end do

contains
    real function f(x)
        real, intent(in) :: x
        ! Transcendental: e^x - 3x
        f = exp(x) - 3.0 * x
    end function f
end program
```

>C++Implementation (*.cpp*)

```cpp
#include <iostream>
#include <vector>
#include <cmath>
#include <functional>
#include <algorithm> // For std::min

/**
 * Incremental Search for root location.
 * Finds intervals [x1, x2] where a sign change occurs.
 */
std::vector<std::pair<double, double>> incremental_search(
    std::function<double(double)> f,
    double a, double b, double step = 0.1)
{
    std::vector<std::pair<double, double>> brackets;
    double x1 = a;
    double f1 = f(x1);

```

```cpp
    while (x1 < b) {
        double x2 = std::min(x1 + step, b);
        double f2 = f(x2);

        // Intermediate Value Theorem Check
        if (f1 * f2 <= 0) {
            brackets.push_back({x1, x2});
        }

        x1 = x2;
        f1 = f2;
    }
    return brackets;
}

int main() {
    // Transcendental function: e^x - 3x
    auto f = [](double x) { return std::exp(x) - 3.0 * x; };

    auto results = incremental_search(f, 0.0, 5.0, 0.2);

    std::cout << "Found " << results.size() << " brackets:" << std::endl;
    for (const auto& b : results) {
        std::cout << "[" << b.first << ", " << b.second << "]" << std::endl
            ;
    }

    return 0;
}
```