Explanation of Iterative Calculation Code for Estimating e^x

Mikolaj Suchon

17/03/2025

Theory

The given Java program numerically approximates the exponential function e^x using the Maclaurin series expansion. This method is a fundamental numerical technique for approximating functions using an infinite series. However, in practical computation, we truncate the series at a certain number of terms based on an error threshold or a maximum iteration limit.

The function e^x can be approximated using Maclaurin series expansion (or Taylor Series around $x = 0$). The Maclaurin series expansion for e^x is given by:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

In this series, each term is of the form $\frac{x^n}{n!}$, and as n increases, the value of each term becomes smaller. Thus, we can stop summing once the additional terms contribute negligibly to the sum.

The error in the approximation can be defined as the difference between successive estimates. The iteration stops when the relative error between two consecutive estimates becomes less than or equal to a specified tolerance, e_s, or when the number of iterations reaches a specified maximum, maxit.

Steps in the Algorithm

The iterative method for approximating e^x involves the following steps:

1. **Initialize the first term**: The first term of the series is 1, so we start with an initial estimate for e^x as $sol = 1$.

2. **Iteratively add terms from the series**: Each term of the series is calculated as $\frac{x^n}{n!}$, and the sum is updated by adding these terms successively.

3. **Calculate the approximate error**: The approximate error e_a is calculated using the formula:

$$e_a = \left| \frac{sol - sol_{\text{old}}}{sol} \right| \times 100$$

   where sol_{old} is the previous value of the estimate.

4. **Stop condition**: The iteration stops when either:

$$e_a \leq e_s \quad \text{or} \quad \text{iterations} \geq \text{maxit}$$

5. **Return the result**: After the stopping condition is met, the final value of the sum sol, the approximate error e_a, and the number of iterations are returned.

Code Explanation

The following Java code implements the described algorithm:

```
public class IterativeCalculation {

    public static double[] IterMeth(double x, double es, int maxit) {
        int iter = 1;
        double sol = 1;
        double ea = 100;

        while (true) {
            double solOld = sol;
            sol += Math.pow(x, iter) / factorial(iter);
            iter++;

            if (sol != 0) {
                ea = Math.abs((sol - solOld) / sol) * 100;
            }

            if (ea <= es || iter >= maxit) {
                break;
            }
        }
        return new double[]{sol, ea, iter};
    }

    private static double factorial(int n) {
        double fact = 1;
        for (int i = 1; i <= n; i++) {
            fact *= i;
        }
        return fact;
    }

    public static void main(String[] args) {
        double x = 2.0;
        double es = 0.01;
        int maxit = 100;

        double[] result = IterMeth(x, es, maxit);
        System.out.println("Estimated Value: " + result[0]);
        System.out.println("Approximate Error: " + result[1] + "%");
        System.out.println("Terms Used: " + (int)result[2]);

        // Compare with built-in function
```

```
        double actualValue = Math.exp(x);
        System.out.println("Actual Value: " + actualValue);
        System.out.println("Absolute Error: " + Math.abs(actualValue - result[0]));
    }
}
```

Key Methods and Variables

- **IterMeth(double x, double es, int maxit)**: This method implements the iterative approach for approximating e^x. It accepts three parameters:
  - x: The input for which we want to calculate e^x.
  - e_s: The desired tolerance for the approximate error.
  - maxit: The maximum number of iterations allowed.

  The method returns an array containing:
  - The estimated value of e^x.
  - The approximate error e_a.
  - The number of terms used in the series.
- **factorial(int n)**: This method calculates the factorial of a given integer n, which is used to compute each term in the series $\frac{x^n}{n!}$.
- **Main Method**: The main method sets up the values for x, e_s, and maxit, calls the `IterMeth` function, and prints the results. It also compares the calculated value with the actual value using the built-in function e^x provided by Java's `Math.exp(x)`.

Result and Comparison

In the main method, after the iterative method has completed:

- The `Estimated Value` is printed, which is the approximation for e^x.
- The `Approximate Error` is printed, showing how close the approximation is to the actual value.
- The `Terms Used` shows how many terms of the series were included before the stopping condition was met.
- The `Actual Value` is the true value of e^x calculated using Java's `Math.exp(x)`.
- The `Absolute Error` is the difference between the actual and the estimated values.

Conclusion

This code implements an iterative method for approximating e^x using its Taylor series. The error decreases as more terms are added, and the algorithm stops when the error is sufficiently small or when the maximum number of iterations is reached. The method is useful for calculating exponential values when a precise value is not needed, and the performance can be controlled through the error tolerance and iteration limit.