|  |  |  |
| --- | --- | --- |
|  | **BAHRIA UNIVERSITY, (Karachi Campus)**  *Department of Software Engineering*  **Assignment 1 - Fall 2023** |  |



COURSE TITLE: **NUMERICAL ANALYSIS** COURSE CODE: **GSC-321**

Class: **BSE-VII (A,B)** Time Allowed:  **1 Week.**

Course Instructor: **Engr. Rahemeen** Max. Marks: **10 marks**

Submission Date: **31-10-2023**

**Question No. 1 [CLO1: 10 Marks]**

1. Define numerical stability in the context of numerical algorithms. How can you determine whether a numerical algorithm is stable?

Numerical stability in the context of numerical algorithms refers to the ability of an algorithm to produce accurate and reliable results even when subjected to small variations or perturbations in the input data. It is a crucial consideration in numerical computations because it ensures that the algorithm behaves predictably and consistently, especially when dealing with real-world data that may have inherent imprecisions or uncertainties.

To determine whether a numerical algorithm is stable, you can consider the following factors:

**Error Analysis:** Perform a thorough error analysis of the algorithm. This involves examining how errors propagate through the algorithm's calculations. A stable algorithm should exhibit bounded error growth, meaning that small errors in the input data do not lead to significant errors in the output.

**Conditioning:** Assess the conditioning of the problem being solved. A well-conditioned problem is one where small change in the input data result in small changes in the output. Stable algorithms are better suited to well-conditioned problems. If the problem is ill-conditioned, it may be more challenging to achieve stability.

**Round-off Errors:** Consider the impact of round-off errors, which occur due to finite precision arithmetic in digital computers. Ensure that the algorithm is designed to minimize or control round-off errors, especially when dealing with iterative or recursive calculations.

**Convergence:** For iterative algorithms, examine the convergence behavior. A stable algorithm should converge to a solution in a predictable and efficient manner. It should not oscillate or diverge when the input data is perturbed slightly.

**Test Cases:** Test the algorithm with a range of input data, including extreme cases and boundary values. Verify that it consistently produces accurate results across a variety of scenarios.

**Analyze Stability Criteria:** Some numerical algorithms have stability criteria derived from mathematical analysis. These criteria provide mathematical conditions that must be met for the algorithm to remain stable. Ensure that these criteria are satisfied.

**Comparison:** Compare the results of the algorithm to analytical solutions or results obtained using other established algorithms. If the algorithm consistently produces similar results, it is an indicator of stability.

**Sensitivity Analysis:** Conduct sensitivity analysis by intentionally introducing small perturbations in the input data and observing the effect on the output. A stable algorithm should demonstrate resilience to such perturbations.

**Empirical Testing:** Finally, conduct extensive empirical testing with real-world data or simulated scenarios to assess the algorithm's performance and stability in practical applications.

1. Discuss the difference between linear convergence and quadratic convergence in the context of iterative numerical methods.

In the context of iterative numerical methods, such as optimization algorithms or solving systems of linear equations, linear convergence and quadratic convergence are two different rates at which an iterative algorithm approaches the solution. These rates are indicators of how quickly the algorithm reduces the error between the current approximation and the true solution.

**Linear Convergence:** Linear convergence means that the error in the approximation decreases linearly with each iteration.

Mathematically, if "e\_k" represents the error at the k-th iteration, then for a linearly convergent method, you would typically see a behavior like this: ek ≈ C⋅ek−1.​

where "C" is a constant, typically less than 1.

In other words, the error is reduced by a fixed fraction with each iteration. Linear convergence is relatively slower compared to quadratic convergence.

Common examples of algorithms with linear convergence include the Jacobi method for solving linear systems of equations and some gradient descent methods for optimization.

**Quadratic Convergence:** Quadratic convergence, on the other hand, implies that the error decreases quadratically with each iteration. Mathematically, for a quadratically convergent method, the error behavior would look like this:

ek​ ≈ C⋅(ek−1​) ^2.

where "C" is a constant. In this case, the error is reduced much faster than with linear convergence. As the number of iterations increases, the error shrinks rapidly, making the algorithm converge to the solution much more quickly.

A classic example of an algorithm with quadratic convergence is the Newton-Raphson method for finding the roots of a function. Newton's method typically converges very quickly when close to the solution.

The main difference between linear convergence and quadratic convergence in iterative numerical methods is the rate at which the error diminishes with each iteration.

**Quadratic convergence** is much faster, while linear convergence progresses at a slower, more constant rate. The choice of method depends on the specific problem and trade-offs between computational cost and convergence speed.

**Linear convergence** is characterized by a linear decrease in error with each iteration, while quadratic convergence features a quadratic decrease in error.