|  |  |  |
| --- | --- | --- |
| **A logo of a university  Description automatically generated** | **BAHRIA UNIVERSITY, (Karachi Campus)**  *Department of Software Engineering*  **Assignment 1 - Fall 2024** |  |

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

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

Course Instructor: **Engr. Rahemeen** Max. Marks: **5 marks**

Submission Date: **18-10-2024**

**Question No. 1 [CLO1: 5 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:**

Numerical stability refers to how an algorithm behaves when it is subjected to small perturbations or rounding errors during computation.

A numerically stable algorithm produces a result that stays near the true solution, even in the presence of minor errors. On the other hand, an unstable algorithm may amplify these small errors, leading to results that deviate significantly from the correct solution.

In practice, numerical stability refers to how well an algorithm keeps errors, such as rounding or floating-point inaccuracies, under control throughout the computation. A stable algorithm ensures that these errors remain small and do not grow large enough to noticeably affect the final outcome.

**Determining if a Numerical Algorithm is stable:**

To determine whether a numerical algorithm is stable, you can use the following approaches:

* **Backward Error Analysis:**  
  This approach involves checking whether the algorithm produces results that can be interpreted as the exact solution to a slightly perturbed version of the original problem. If the errors in the input lead to small changes in the output, the algorithm is stable.
* **Sensitivity to Input Errors (Condition Number):**  
  The condition number of a problem measures how much the output is affected by small changes in the input. Even if the algorithm is stable, the problem itself might be ill-conditioned, meaning small input errors could result in large output errors. A numerically stable algorithm will generally have outputs proportional to the condition number of the problem.
* **Error Propagation Testing:**  
  Implementing the algorithm with various levels of precision (e.g., single, double, or higher floating-point precision) and observing how the errors propagate. If the results vary wildly as precision changes, it suggests instability.
* **Round-off Error Growth:**  
  Monitor the round-off errors through the steps of the algorithm. A stable algorithm will not allow these round-off errors to grow uncontrollably. Analyzing the floating-point operations used by the algorithm can help assess this.
* **Empirical Testing:**  
  Run the algorithm on a wide range of test cases, particularly with inputs designed to stress-test the algorithm’s numerical properties (e.g., very large or small numbers). If the algorithm produces reliable and accurate results under these conditions, it is likely numerically stable.

**CONCLUSION:**

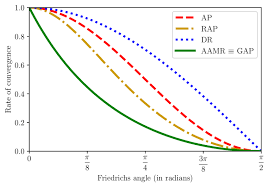
In summary, numerical stability ensures that an algorithm performs reliably even in the presence of small computational errors, and this can be assessed through techniques such as backward error analysis, condition number analysis, error propagation tests, and empirical validation.

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

**Answer:**

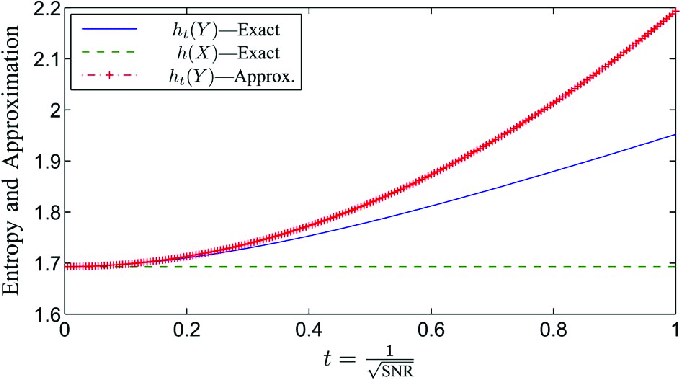
In iterative numerical methods, the **rate of convergence** describes how quickly the error decreases as the iteration progresses. Two common types of convergence are **linear convergence** and **quadratic convergence**, which differ in their rates of error reduction:

**Linear Convergence:**



* **Definition**: A sequence {xn}\{x\_n\}{xn​} converges to a solution x∗x^\*x∗ with *linear convergence* if the error en=∣xn−x∗∣e\_n = |x\_n - x^\*|en​=∣xn​−x∗∣ decreases in proportion to the previous error as n→∞n \to \inftyn→∞. en+1≤C⋅enfor some constant C∈(0,1).e\_{n+1} \leq C \cdot e\_n \quad \text{for some constant } C \in (0, 1).en+1​≤C⋅en​for some constant C∈(0,1).
* **Error behavior**: The error is reduced by a constant factor in each iteration. For example, if the error is halved in each iteration (C=0.5C = 0.5C=0.5), then the sequence is linearly converging.
* **Convergence speed**: Linear convergence is relatively slow. While the error decreases consistently, it does so at a constant rate that doesn't accelerate over time.
* **Example**: The **fixed-point iteration** method typically exhibits linear convergence when the initial guess is close to the solution.

**Quadratic Convergence:**



* **Definition**: A sequence {xn}\{x\_n\}{xn​} converges to x∗x^\*x∗ with *quadratic convergence* if the error satisfies: en+1≤C⋅en2for some constant C>0.e\_{n+1} \leq C \cdot e\_n^2 \quad \text{for some constant } C > 0.en+1​≤C⋅en2​for some constant C>0.
* **Error behavior**: The error decreases quadratically with each iteration, meaning the number of correct digits approximately doubles in each iteration.
* **Convergence speed**: Quadratic convergence is much faster than linear convergence. As the error decreases, subsequent iterations become exponentially more accurate, significantly accelerating convergence.
* **Example**: The **Newton-Raphson method** for solving nonlinear equations often exhibits quadratic convergence when the initial guess is sufficiently close to the true root.

**KEY DIFFERENCE**

|  |  |
| --- | --- |
| **Linear convergence** | **Quadratic convergence** |
| In linear convergence, the error reduces proportionally to the previous error. | in quadratic convergence, the error decreases in proportion to the square of the previous error. |
| linear convergence is steady but slower. | Quadratic convergence is faster, especially as the iterations progress. |

**CONCLUSION:**

In summary, **quadratic convergence** provides a much faster approach to reaching a solution compared to **linear convergence**, especially as iterations increase, but it often requires stronger conditions, such as a good initial guess or higher computational cost per iteration.

**REFERENCES**:  
  
<https://en.wikipedia.org/wiki/Rate_of_convergence>  
<https://planetmath.org/linearconvergence>

<https://www.sciencedirect.com/topics/mathematics/linear-convergence>

<https://sites.math.washington.edu/~morrow/464_19/quadratic%20convergence.pdf>

<https://www.sciencedirect.com/topics/engineering/quadratic-convergence>