

**Discovering Knowledge**

**COURSE: GSL 321**

**NUMERICAL ANALYSIS**

**PROJECT REPORT**

**CLASS: BSE – 7B (FALL - 2023)**

**Implementation of Gauss Jacobi, Gauss Siedel and Successive Over Relaxation Method**

**Group Members**

|  |  |
| --- | --- |
| **Student Name** | **Enrollment#** |
| Muhammad Amjad | 02-131202-041 |
| Jamshed Ali | 02-131202-008 |
| Abu Hurara | 02-131202-016 |

**Submitted to:**

Course Instructor: Engr. Rahemeen

Lab Instructor: Engr. Noor us Sabah

**Department of Software Engineering**

**BAHRIA UNIVERSITY KARACHI CAMPUS**

**Abstract**

This project involves implementing Gauss-Jacobi, Gauss-Seidel, and Successive Over-Relaxation methods in Python to solve linear systems. The focus is on evaluating their strengths and weaknesses in terms of convergence, stability, and efficiency. Utilizing Matplotlib for visualization, the project includes a parameter sensitivity study, emphasizing the impact of relaxation parameters in the SOR method. Real-world applications will be simulated to assess practical performance. The goal is to provide practical insights into optimizing the use of these iterative methods for solving large-scale linear systems.

TABLE OF Contents

[1. Introduction 4](#_Toc155309748)

[2. Problem Statement 4](#_Toc155309749)

[3. Proposed Solution 4](#_Toc155309750)

[3.1. Features of the project 4](#_Toc155309751)

[3.2. Methodology 4](#_Toc155309752)

[3.3. Technologies 5](#_Toc155309753)

[4. Project Scope 5](#_Toc155309754)

[5. Module Distribution 5](#_Toc155309755)

[6. Code 6](#_Toc155309756)

[7. Conclusion 10](#_Toc155309757)

[8. References 10](#_Toc155309758)

# Introduction

Linear systems, represented as (Ax = b), play a pivotal role in mathematical modeling across scientific and engineering domains. This project delves into the implementation and comparison of three iterative methods—Gauss-Jacobi, Gauss-Seidel, and Successive Over-Relaxation (SOR). These methods are essential for approximating solutions to large-scale linear systems encountered in applications such as structural analysis and fluid dynamics. The Gauss-Jacobi method, renowned for its simplicity, is augmented by the Gauss-Seidel method, which updates variables more dynamically. Successive Over-Relaxation (SOR) further refines convergence by introducing a relaxation parameter. This project aims to not only implement these methods but also analyze their convergence rates, stability, and computational efficiency, providing valuable insights for optimizing solutions to practical problems.

# Problem Statement

Effectively solving large linear systems is crucial in various fields, and iterative methods like Gauss-Jacobi, Gauss-Seidel, and Successive Over-Relaxation (SOR) offer solutions. However, understanding their behavior and choosing the right method for specific scenarios remains a challenge. This project aims to investigate the convergence, stability, and efficiency of these methods in different situations, addressing key questions such as their performance in diverse linear systems and the impact of parameters.

# Proposed Solution

## Features of the project

The project will implement and assess the Gauss-Jacobi, Gauss-Seidel, and Successive Over-Relaxation (SOR) methods in Python, focusing on understanding their strengths and weaknesses. Key features include:

* **Method Implementation:** Code development in Python for the three iterative methods.
* **Convergence Analysis:** Systematic evaluation of convergence behavior under varying conditions to identify strengths and weaknesses.
* **Visualization:** Use of Matplotlib for clear visualization of convergence patterns and other relevant metrics.

## Methodology

* **Implementation:** Code development in Python, ensuring simplicity and clarity in the implementation of the Gauss-Jacobi, Gauss-Seidel, and SOR methods.
* **Convergence Analysis:** Systematic testing and analysis of convergence rates for different types of linear systems.
* **Visualization:** Clear graphical representation of convergence behavior and other performance metrics using Matplotlib.

## Technologies

* **Programming Language:** Python for its simplicity and readability.
* **Visualization:** Matplotlib for creating visualizations that aid in understanding the strengths and weaknesses of the implemented methods.

# Project Scope

The scope of this project is defined by its focus on implementing and evaluating the Gauss-Jacobi, Gauss-Seidel, and Successive Over-Relaxation (SOR) methods for solving linear systems using Python. The project's boundaries and objectives include:

* **Implementation:** Develop Python code for the three iterative methods with a primary emphasis on simplicity, efficiency, and clarity.
* **Method Evaluation:** Conduct a comprehensive analysis of the implemented methods, assessing their strengths and weaknesses in terms of convergence rates, stability, and computational efficiency.
* **Parameter Sensitivity:** Investigate the impact of relaxation parameters, especially in the SOR method, on the convergence behavior to provide insights into optimal parameter selection.
* **Visualization:** Utilize Matplotlib for visual representation of convergence patterns and other relevant metrics, aiding in the interpretation of method performance.
* **Real-world Applications:** Apply the implemented methods to practical linear systems inspired by real-world scenarios in engineering and science to assess their applicability and performance.

# Module Distribution

|  |  |  |
| --- | --- | --- |
| **Name** | **Enrollment #** | **Module** |
| Muhammad Amjad | 02-131202-041 | SOR Method, Visualization |
| Jamshed Ali | 02-131202-008 | Guass Siedel Iteration Method |
| Abu Hurara | 02-131202-016 | Guass Jacobi Method |

# Code

a11 = float(input("Enter the value of a11: "))

a12 = float(input("Enter the value of a12: "))

a13 = float(input("Enter the value of a13: "))

b1 = float(input("Enter the value of b1: "))

a21 = float(input("Enter the value of a21: "))

a22 = float(input("Enter the value of a22: "))

a23 = float(input("Enter the value of a23: "))

b2 = float(input("Enter the value of b2: "))

a31 = float(input("Enter the value of a31: "))

a32 = float(input("Enter the value of a32: "))

a33 = float(input("Enter the value of a33: "))

b3 = float(input("Enter the value of b3: "))

e=float(input("Enter Tolerance Error: "))

f1 = lambda x,y,z: (b1+a12\*y-a13\*z)/a11

f2 = lambda x,y,z: (b2+a21\*x-a23\*z)/a22

f3 = lambda x,y,z: (b3-a31\*x-a32\*y)/a33

## Gauss Jacobi:

if a11<(a12+a13) or a22<(a21+a23) or a33<(a31+a32):

  print("These linear equations can't be solved by Guass Siedel or Guass Jacobi")

else:

  x0 = 0

  y0 = 0

  z0 = 0

  count = 1

  print('\nCount\tx\ty\tz\n')

  condition = True

  iteration\_counts = []

  errors = []

  while condition:

      x1 = f1(x0,y0,z0)

      y1 = f2(x0,y0,z0)

      z1 = f3(x0,y0,z0)

      print('%d\t%0.4f\t%0.4f\t%0.4f\n' %(count, x1,y1,z1))

      e1 = abs(x0-x1);

      e2 = abs(y0-y1);

      e3 = abs(z0-z1);

      iteration\_counts.append(count)

      errors.append(max(e1, e2, e3))

      count += 1

      x0 = x1

      y0 = y1

      z0 = z1

      condition = e1>e and e2>e and e3>e

  print('\nSolution: x=%0.3f, y=%0.3f and z = %0.3f\n'% (x1,y1,z1))

import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))

plt.bar(iteration\_counts, errors, color='blue', alpha=0.7)

plt.xlabel('Iteration Count')

plt.ylabel('Error')

plt.title('Convergence of Gauss Jacobi Iteration')

plt.grid(True)

plt.show()

plt.figure(figsize=(10, 6))

plt.semilogy(iteration\_counts, errors, marker='o', linestyle='-')

plt.xlabel('Iteration Count')

plt.ylabel('Error')

plt.title('Convergence Analysis - Gauss-Jacobi Method')

plt.grid(True)

plt.show()

## Gauss Siedel

if a11<(a12+a13) or a22<(a21+a23) or a33<(a31+a32):

  print("These linear equations can't be solved by Guass Siedel or Guass Jacobi")

else:

  x0 = 0

  y0 = 0

  z0 = 0

  count = 1

  print('\nCount\tx\ty\tz\n')

  condition = True

  iteration\_counts = []

  errors = []

  while condition:

      x1 = f1(x0,y0,z0)

      y1 = f2(x1,y0,z0)

      z1 = f3(x1,y1,z0)

      print('%d\t%0.4f\t%0.4f\t%0.4f\n' %(count, x1,y1,z1))

      e1 = abs(x0-x1);

      e2 = abs(y0-y1);

      e3 = abs(z0-z1);

      iteration\_counts.append(count)

      errors.append(max(e1, e2, e3))

      count += 1

      x0 = x1

      y0 = y1

      z0 = z1

      condition = e1>e and e2>e and e3>e

  print('\nSolution: x=%0.3f, y=%0.3f and z = %0.3f\n'% (x1,y1,z1))

import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))

plt.bar(iteration\_counts, errors, color='blue', alpha=0.7)

plt.xlabel('Iteration Count')

plt.ylabel('Error')

plt.title('Convergence of Gauss Siedel Iteration')

plt.grid(True)

plt.show()

plt.figure(figsize=(10, 6))

plt.semilogy(iteration\_counts, errors, marker='o', linestyle='-')

plt.xlabel('Iteration Count')

plt.ylabel('Error')

plt.title('Convergence Analysis - Gauss-Siedel Method')

plt.grid(True)

plt.show()

## Successive Over Relaxation

if a11<(a12+a13) or a22<(a21+a23) or a33<(a31+a32):

  print("These linear equations can't be solved by Guass Siedel or Guass Jacobi or SOR Method")

else:

  x0 = 0

  y0 = 0

  z0 = 0

  count = 1

  w = float(input("Enter relaxation factor: "))

  print('\nCount\tx\ty\tz\n')

  iteration\_counts = []

  errors = []

  condition = True

  while condition:

      x1 = (1-w) \* x0 + w \* f1(x0,y0,z0)

      y1 = (1-w) \* y0 + w \* f2(x1,y0,z0)

      z1 = (1-w) \* z0 + w \* f3(x1,y1,z0)

      print('%d\t%0.4f\t%0.4f\t%0.4f\n' %(count, x1,y1,z1))

      e1 = abs(x0-x1);

      e2 = abs(y0-y1);

      e3 = abs(z0-z1);

      iteration\_counts.append(count)

      errors.append(max(e1, e2, e3))

      count += 1

      x0 = x1

      y0 = y1

      z0 = z1

      condition = e1>e and e2>e and e3>e

  print('\nSolution: x = %0.3f, y = %0.3f and z = %0.3f\n'% (x1,y1,z1))

import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))

plt.bar(iteration\_counts, errors, color='blue', alpha=0.7)

plt.xlabel('Iteration Count')

plt.ylabel('Error')

plt.title('Convergence of SOR Method')

plt.grid(True)

plt.show()

plt.figure(figsize=(10, 6))

plt.semilogy(iteration\_counts, errors, marker='o', linestyle='-')

plt.xlabel('Iteration Count')

plt.ylabel('Error')

plt.title('Convergence Analysis - SOR Method')

plt.grid(True)

plt.show()

plt.bar(mathod\_names, max\_iterations, color='blue')

plt.xlabel('Method Names')

plt.ylabel('Max Iterations')

plt.title('Max Iterations for Different Methods')

plt.show()

## Successive Over Relaxation

plt.bar(mathod\_names, max\_iterations, color='blue')

plt.xlabel('Method Names')

plt.ylabel('Max Iterations')

plt.title('Max Iterations for Different Methods')

plt.show()

# Conclusion

In our numerical analysis lab project, we implemented Gauss-Seidel, Gauss-Jacobi, and Successive Overrelaxation methods to solve an equation. Through thorough analysis and computational experiments, we compared their convergence and efficiency. Graphs were created to visually represent the convergence behavior. The project enhanced our understanding of these iterative methods, providing both theoretical insights and practical experience in their application to complex equations.

# References

* Wolfram MathWorld. "Successive Overrelaxation Method." [Online]. Available: <https://mathworld.wolfram.com/SuccessiveOverrelaxationMethod.html>.
* Byju's. "Iterative Methods - Gauss Seidel and Jacobi." [Online]. Available: <https://byjus.com/maths/iterative-methods-gauss-seidel-and-jacobi/>.
* LibreTexts. "Gauss-Seidel Method." [Online]. Available: <https://math.libretexts.org/Bookshelves/Linear_Algebra/Introduction_to_Matrix_Algebra_(Kaw)/01%3A_Chapters/1.08%3A_Gauss-Seidel_Method>.
* Mathematical Association of America. "Iterative Methods for Solving Ax=b: Jacobi's Method." [Online]. Available: <https://maa.org/press/periodicals/loci/joma/iterative-methods-for-solving-iaxi-ibi-jacobis-method>.