暨南大学本科实验报告专用纸

课程名称 数值计算实验 成绩评定

实验项目名称 Computing Problems 指导老师 Your Director

实验项目编号 080812345601 实验项目类型 实验地点 N117

学生姓名 学生A，学生B 学号 2020101234，2019051234

学院 信息科学技术学院 系 计算机科学 专业 计算机科学与技术

实验日期 2021年3月1日～2021年3月1日 温度 °C 湿度

I. Problem

Let A be the 1000 × 1000 matrix with entries $A(i,i) = i, A(i,i+1) = A(i+1,i) = \frac{1}{2}, A(i,i+2) = A(i+2,i) = \frac{1}{2}$ for all i that fit within the matrix.

- Solve the system with $Ax = [1, 1, \cdot \cdot \cdot, 1]^T$ by the following methods in 15 steps:
    1. The Jacobi Method;
    2. The Gauss-Seidel Method;
    3. SOR with $\omega = 1.1$;
    4. The Conjugate Gradient Method;
    5. The Conjugate Gradient Method with Jacobi preconditioner.
- Report the errors of every step for each method.

II. Algorithm Summary

1. The Jacobi Method

The Jacobi Method is a form of fixed-point iteration for a system of equations.

- Let D denote the main diagonal of A, L denote the lower triangle of A (entries below the main diagonal), and U denote the upper triangle (entries above the main diagonal).
- Then $A = L + D + U$ and the equation to be solved is $Lx + Dx + Ux = b$. Note that this use of L and U differs from the use in the LU factorization, since all diagonal entries of this L and U are zero.

III. Experimental Summary

In this experiment, we use five methods to solve the system with $Ax = [1, 1, ..., 1]^T$ in 15 steps. We find that the Gauss-Seidel method converges the fastest and the error of the result is the smallest. Next is the Conjugate Gradient method with Jacobi preconditioner, then is the SOR with $\omega = 1.1$, next is the Jacobi method. The worst is the Conjugate Gradient method which is far inferior to other methods.

We notice that matrix A is not A strictly diagonally dominant matrix, but A weakly diagonally dominant matrix. The applicable condition of the first three methods (Jacobi, GAuss-Seidel, SOR) is strictly diagonally dominant matrix, so the first three methods are not suitable for this experiment.

A Appendix: Source Code

1. Problem.py

```
import numpy as np

print("Hello")

def incmatrix(genl1,genl2):
    m = len(genl1)
    n = len(genl2)
    M = None #to become the incidence matrix
    VT = np.zeros((n*m,1), int)  #dummy variable

...
    return M
```

2. Problem.m

```
for n = 1:2
    for m = 1:3
        fprintf('n = %3u m = %3u \r', n, m)
        % This is a comment
    end
end
```

3. Problem.c

```
#include <stdio.h>
int main(){
    printf("Hello world");
    // This is a comment.
}
```