# 计算方法 B 课程实验报告 4

**姓名：**章东泉　　**学号：**PB17121706

## 问题一　迭代法解线性代数方程组

### Part1 计算结果

由于题目禁止对矩阵求逆，于是使用分量形式进行迭代。

Jacobi 迭代的结果如下表.

表 1-1　Jacobi 迭代步数与最终结果

| 迭代步数/结果 | 取值 |
|---|---|
| 迭代步数 | 333 |
| $x_1$ | 9.9999907919e+000 |
| $x_2$ | 1.7999982330e+001 |
| $x_3$ | 2.3999975299e+001 |
| $x_4$ | 2.7999970270e+001 |
| $x_5$ | 2.9999967649e+001 |
| $x_6$ | 2.9999967649e+001 |
| $x_7$ | 2.7999970270e+001 |
| $x_8$ | 2.3999975299e+001 |
| $x_9$ | 1.7999982330e+001 |
| $x_{10}$ | 1.7999982330e+001 |

Gauss-Seidel 迭代的结果如下表.

表 1-2　Gauss-Seidel 迭代步数与最终结果

| 迭代步数/结果 | 取值 |
|---|---|
| 迭代步数 | 175 |
| $x_1$ | 9.9999943683e+000 |
| $x_2$ | 1.7999989631e+001 |
| $x_3$ | 2.3999986092e+001 |
| $x_4$ | 2.7999983938e+001 |
| $x_5$ | 2.9999983230e+001 |
| $x_6$ | 2.9999983909e+001 |
| $x_7$ | 2.7999985812e+001 |
| $x_8$ | 2.3999988690e+001 |
| $x_9$ | 1.7999992237e+001 |
| $x_{10}$ | 9.9999961183e+000 |

## Part2 算法分析

两种迭代计算都使用了对应的分量形式，其中：

Jacobi 迭代的分量形式为：

$$x_i^{k+1} = -\frac{1}{a_{ii}}(a_{i1}x_1^{(k)} + \ldots + a_{ii-1}x_{i-1}^{(k)} + a_{ii+1}x_{i+1}^{(k)} + \ldots a_{in}x_n^{(k)} - b_i)$$

Gauss-Seidel 迭代的分量形式为：

$$x_i^{k+1} = -\frac{1}{a_{ii}}(a_{i1}x_1^{(k+1)} + \ldots + a_{ii-1}x_{i-1}^{(k+1)} + a_{ii+1}x_{i+1}^{(k)} + \ldots a_{in}x_n^{(k)} - b_i)$$

迭代到无穷范数$\left\|x^{(k+1)}-x^{(k)}\right\|_{\infty}<\varepsilon$时停止迭代。

## Part3 结果分析

显然，在本例中 175 步收敛的 Gauss-Seidel 迭代收敛速度要快于 333 步收敛的 Jacobi 迭代。虽然实际上收敛快慢由谱半径决定，但大多数情况下 Gauss-Seidel 迭代的收敛速度都要快于 Jacobi 迭代。

## Part4 实验小结和代码展示

实验中使用两种迭代方法求解线性代数方程组，由于许多计算问题都可以转为线性代数方程组解决，故此方向研究在数值计算领域有着重要的地位。

图 4-1 Jacobi 单一迭代值计算

```
double Jacobi_iteration(int n)
{
    int i = 0;
    double result = 0.0;
    while(i < size){
        result += A[n][i] * old_X[i];
        i++;
    }
    result = -(result - b[n])/A[n][n] + old_X[n];
    return result;
}
```

图 4-2　Gauss-Seidel 单一迭代值计算

```
double Gauss_iteration(int n)
{
    int i = 0;
    double result = 0.0;
    while(i < n){
        result += A[n][i] * new_X[i];
        i++;
    }
    i++;
    while(i < size){
        result += A[n][i] * old_X[i];
        i++;
    }
    result = -(result - b[n])/A[n][n];
    return result;
}
```

图 4-3　Gauss-Seidel 迭代

```
while(infinite_norm() >= epsilon){
    i = 0;
    while(i < size){
        old_X[i] = new_X[i];
        i++;
    }
    i = 0;
    while(i < size){
        new_X[i] = Gauss_iteration(i);
        i++;
    }
    round++;
    printf("Round %d:\n",round);
    i = 0;
    while(i < size){
        printf("%.10e\n", new_X[i]);
        i++;
    }
    printf("\n");
}
```

图 4-4　代码原始输出

```
Round 175:
9.9999943683e+000
1.7999989631e+001
2.3999986092e+001
2.7999983938e+001
2.9999983230e+001
2.9999983909e+001
2.7999985812e+001
2.3999988690e+001
1.7999992237e+001
9.9999961183e+000
```