Lab6 The comparison between Jacobi iteration and Gauss-Seidel iteration

Sheng Zhang
2016310200210
2nd class of information and computing science

1 Lab purpose

Solve the equation utilizing Jacobi and Gauss-Seidel iteration respectively. For the purpose of reproducibility, the whole codes are available in Github repository[1].

2 Lab preparation

Read section 6.2 of the textbook «numerical analysis».

3 Lab requirements

C language implementation of Jacobi and Gauss-Seidel iteration.

3.1 Definition of function interface

```
int Jacobi( int n, double a[][MAX_SIZE], double b[], double x[], double
    TOL, int MAXN );
int Gauss_Seidel( int n, double a[][MAX_SIZE], double b[], double x[],
   double TOL, int MAXN );
```

The definitions of two function are the same. n denotes the dimension of matrix a, MAX-SIZE denotes the maximal dimension of matrix, b denotes the constant vector of equation. At the beginning of iteration, x denotes initial vector and after iteration, it denotes the solution of equation. TOL denotes error upper-bound. MAXN denotes the maximal times of iteration.

3.2 Data requirements

Input:
3
2 -1 1 -1
2 2 2 4
-1 -1 2 -5
0.000000001 1000
3
1 2 -2 7
1 1 1 2
2 2 1 5
0.000000001 1000
5
2 1 0 0 0 1
1 2 1 0 0 1
0 1 2 1 0 1
0 0 1 2 1 1
0 0 0 1 2 1
0.000000001 100

Output:

[1] https://github.com/ZZshengyeah/Numerical-Analysis

Result of Jacobi method:
No convergence.
Result of Gauss-Seidel method:
no-iteration = 37
1.00000000
2.00000000
-1.00000000

Result of Jacobi method:
no-iteration = 232
1.00000000
2.00000000
-1.00000000
Result of Gauss-Seidel method:
No convergence.

Result of Jacobi method:
Maximum number of iterations exceeded.
Result of Gauss-Seidel method:
no-iteration = 65
0.50000000
0.00000000
0.50000000
0.00000000
0.50000000

4 Detail of Jacobi and Gauss-Seidel iteration

The computing formulas of Jacobi iteration are as follows.

$$\begin{cases} \mathbf{x^{(0)}} = (x_1^{(0)}, x_2^{(0)}, ..., x_n^{(0)})^T \\ x_i^{(k+1)} = (b_i - \sum\limits_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^{(k)})/a_{ii} \\ i = 1, 2, ..n; k = 0, 1, ... \end{cases}$$

Similarly, the computing formulas of Gauss-Seidel iteration are as follows.

$$\begin{cases} \mathbf{x^{(0)}} = (x_1^{(0)}, x_2^{(0)}, ..., x_n^{(0)})^T \\ x_i^{(k+1)} = (b_i - \sum\limits_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum\limits_{j=i}^{n} a_{ij} x_j^{(k)})/a_{ii} \\ i = 1, 2, ..n; k = 0, 1, ... \end{cases}$$

5 Code analysis and thinking

As far as I am concerned, the implementation of these two iterations is not difficult, but we need to pay attention to the part of column principle elimination. At the beginning, I didn't do the column principle elimination, and found the number of the second iteration is not the same as standard solution. Then after implementation of column principle elimination, it was right. Be careful in programming is important. And the code result is shown in Figure 1.

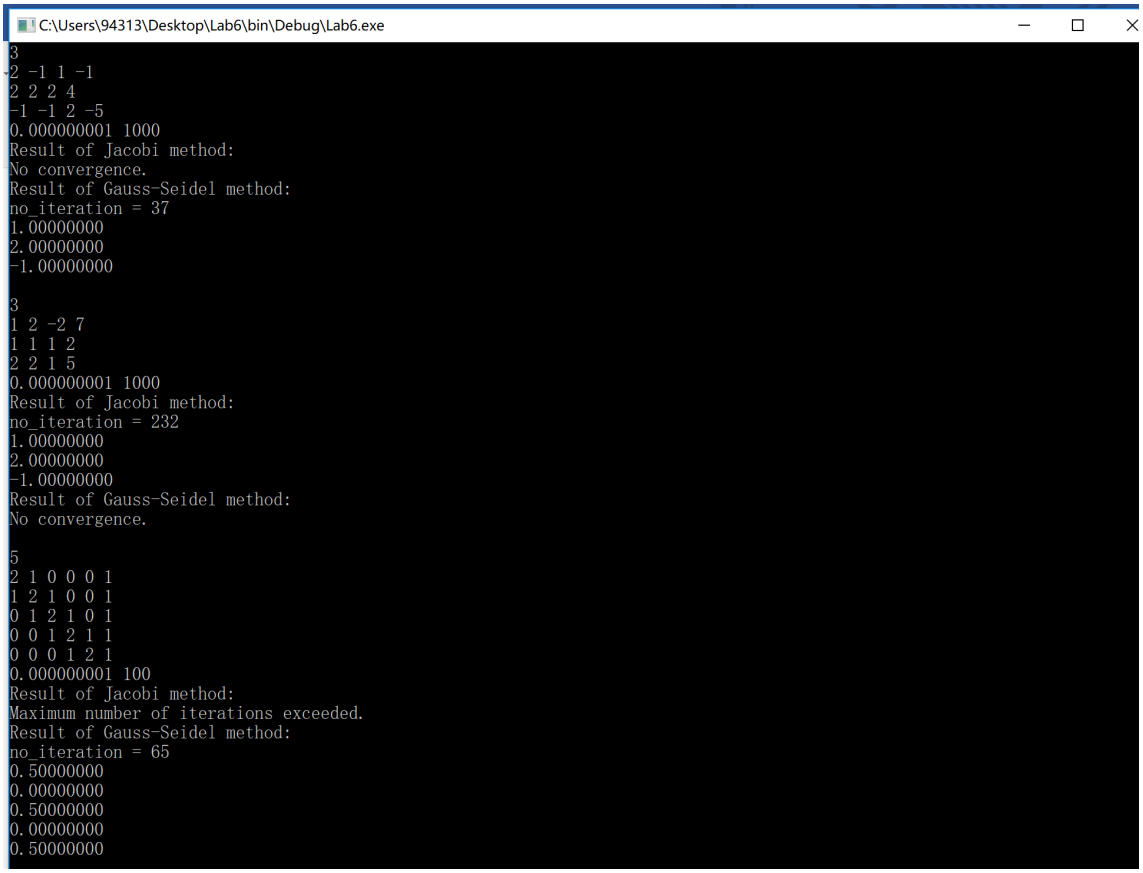

Figure 1: The results of Jacobi and Gauss-Seidel iteration.

6 Code sharing

```c
#include<stdio.h>
#include<math.h>
#define MAX_SIZE 100
#define bound pow(2, 127)
#define ZERO 0.000000001

double norm(int n, double x[], double y[])
{
    double max=0;
    for(int i=0;i<n;i++)
    {
        double val=fabs(x[i]-y[i]);
        if(val>max)
            max=val;
    }
    return max;
}
void swap(int i,int j, int n, double a[][MAX_SIZE], double b[])
{
    double temp;
    for(int count=0;count<n;count++)
    {
        temp = a[i][count];
        a[i][count] = a[j][count];
        a[j][count] = temp;
        temp = b[i];
        b[i] = b[j];
        b[j] = temp;
    }
}
```

```
void change(int n, double a[][MAX_SIZE], double b[])
{
    for(int i=0;i<n;i++)
    {
        double max = a[i][i];
        for(int j=i+1;j<n;j++)
        {
            if(fabs(a[j][i])>fabs(max))
                swap(i,j,n,a,b);
        }
    }
}
int convergence(int n, double x[])
{
    for(int i=0;i<n;i++)
    {
        if(x[i]>bound || x[i]<-bound)
            return 0;
    }
    return 1;
}
int test(int n, double a[][MAX_SIZE])
{
    for(int i=0;i<n;i++)
    {
        int count=0;
        for(int j=0;j<n;j++)
        {
            if(a[j][i]==0)
                count++;
        }
        if(count == n)
            return 0;
    }
    return 1;
}
int Jacobi( int n, double a[][MAX_SIZE], double b[], double x[], double
    TOL, int MAXN )
{
    change(n,a,b);
    if(test(n, a)==0)
        return -1;
    double update_x[n];
    int k=1;
    while(1)
    {
        for(int i=0;i<n;i++)
        {
            double sum=0;
            for(int j=0;j<n;j++)
            {
                if(j==i)
                    continue;
                sum += a[i][j]*x[j];
            }
            update_x[i] = (b[i]-sum)/a[i][i];
        }
        if(norm(n,x,update_x)<=TOL)
        {
            for(int i=0;i<n;i++)
                x[i]=update_x[i];
            break;
        }
```

```
        if(convergence(n,update_x) == 0)
            return -2;
        for(int i=0;i<n;i++)
            x[i]=update_x[i];
        if(k>MAXN)
            return 0;
        k++;
    }
    return k;
}
int Gauss_Seidel ( int n, double a[][MAX_SIZE], double b[], double x[],
    double TOL, int MAXN )
{
    if(test(n, a)==0)
        return -1;
    double update_x[n];
    int k=1;
    while(1)
    {
        for(int i=0;i<n;i++)
        {
            double sum1=0,sum2=0;
            for(int j=0;j<=i-1;j++)
                sum1 += a[i][j]*update_x[j];
            for(int j=i+1;j<n;j++)
                sum2 += a[i][j]*x[j];
            update_x[i] = (b[i]-sum1-sum2)/a[i][i];
        }
        if(norm(n,x,update_x)<=TOL)
        {
            for(int i=0;i<n;i++)
                x[i]=update_x[i];
            break;
        }
        if(convergence(n,update_x) == 0)
            return -2;
        for(int i=0;i<n;i++)
            x[i]=update_x[i];
        if(k>MAXN)
            return 0;
        k++;
    }
    return k;
}
int main()
{
  int n, MAXN, i, j, k;
  double a[MAX_SIZE][MAX_SIZE], b[MAX_SIZE], x[MAX_SIZE];
  double TOL;

  while ( scanf("%d", &n) != EOF ) {
    for ( i=0; i<n; i++ ) {
      for ( j=0; j<n; j++ )
        scanf("%lf", &a[i][j]);
      scanf("%lf", &b[i]);
    }
    scanf("%lf %d", &TOL, &MAXN);
    printf("Result of Jacobi method:\n");
    for ( i=0; i<n; i++ )
      x[i] = 0.0;
    k = Jacobi( n, a, b, x, TOL, MAXN );
    switch ( k ) {
      case -2:
```

```
        printf("No convergence.\n");
        break;
      case -1:
        printf("Matrix has a zero column. No unique solution exists.\n"
            );
        break;
      case 0:
        printf("Maximum number of iterations exceeded.\n");
        break;
      default:
        printf("no_iteration = %d\n", k);
        for ( j=0; j<n; j++ )
          printf("%.8f\n", x[j]);
        break;
    }
    printf("Result of Gauss-Seidel method:\n");
    for ( i=0; i<n; i++ )
      x[i] = 0.0;
    k = Gauss_Seidel( n, a, b, x, TOL, MAXN );
    switch ( k ) {
      case -2:
        printf("No convergence.\n");
        break;
      case -1:
        printf("Matrix has a zero column. No unique solution exists.\n"
            );
        break;
      case 0:
        printf("Maximum number of iterations exceeded.\n");
        break;
      default:
        printf("no_iteration = %d\n", k);
        for ( j=0; j<n; j++ )
          printf("%.8f\n", x[j]);
        break;
    }
    printf("\n");
  }

  return 0;
}
```