# 实习七实习报告

# 实习要求

**根据要求，通过题目中所给的值构建出A矩阵，通过雅阁比方法，高斯赛德尔方法和SOR方法迭代求解，而AX=B的方程中，B的值使用a\*h\*h替代。当完成这项工作之后，对于lamuda的值进行改变，继续比较结果。**

# 算法描述

1. **构建A矩阵和对应的B矩阵向量。**
2. **对应的雅阁比方法求解，输出结果，注意其中i！=j的条件限制。**
3. **对应的高斯赛德尔方法求解，输出结果，注意用到了不同的阶进行计算。**
4. **对应的SOR方法计算，首先对于w赋值，本实验中使用的是0.5，因为其取值居中，适应性强，比较保险。**

# 程序清单

注意countd和lamuda分别对应迭代次数以及需要修改的参数，可以自行修改值。

#include<iostream>

#include<cmath>

#include<math.h>

using namespace std;

int n=100;

double h=1.0/n;

double lamuda=1.0;

double a=0.5;

double A[105][105];

double b[105];

double x[100005][105];//解

int countd;

void goujian()

{

for(int i=1;i<=100;i++)

{

b[i]=a\*h\*h;

}

for(int i=1;i<=100;i++)

{

A[i][i]=-(2\*lamuda+h);

}

for(int i=2;i<=100;i++)

{

A[i][i-1]=lamuda;

}

for(int i=2;i<=100;i++)

{

A[i-1][i]=lamuda+h;

}

}

void jagoubi()

{

for(int i=0;i<=countd;i++)

{

for(int k=0;k<=n;k++)

{

x[i][k]=0.0;

}

}

for(int k=0;k<countd;k++)

{

for(int i=1;i<=n;i++)

{

double sum1=0.0;

for(int j=1;j<=n;j++)

{

if(i!=j)

{

sum1=sum1+A[i][j]\*x[k][j];

}

}

x[k+1][i]=(b[i]-sum1)/A[i][i];

}

}

cout<<"雅阁比开始了！"<<endl;

for(int i=1;i<=100;i++)

{

cout.precision(4);

cout<<x[countd][i]<<endl;

}

cout<<endl;

}

void gauss()

{

for(int i=0;i<=countd;i++)

{

for(int k=0;k<=n;k++)

{

x[i][k]=0.0;

}

}

for(int k=0;k<countd;k++)

{

for(int i=1;i<=n;i++)

{

double sum1=0.0,sum2=0.0;

for(int j=1;j<=i-1;j++)

{

sum1=sum1+A[i][j]\*x[k+1][j];

}

for(int j=i+1;j<=n;j++)

{

sum2=sum2+A[i][j]\*x[k][j];

}

x[k+1][i]=(b[i]-sum1-sum2)/A[i][i];

}

}

cout<<"高斯开始了！"<<endl;

for(int i=1;i<=100;i++)

{

cout.precision(4);

cout<<x[countd][i]<<endl;

}

cout<<endl;

}

void sor()

{

double w=0.5;

for(int i=0;i<=countd;i++)

{

for(int k=0;k<=n;k++)

{

x[i][k]=0.0;

}

}

for(int k=0;k<countd;k++)

{

for(int i=1;i<=n;i++)

{

double sum1=0.0,sum2=0.0;

for(int j=1;j<=i-1;j++)

{

sum1=sum1+A[i][j]\*x[k+1][j];

}

for(int j=i;j<=n;j++)

{

sum2=sum2+A[i][j]\*x[k][j];

}

x[k+1][i]=x[k][i]+w\*(b[i]-sum1-sum2)/A[i][i];

}

}

cout<<"SOR开始了！"<<endl;

for(int i=1;i<=100;i++)

{

cout.precision(4);

cout<<x[countd][i]<<endl;

}

cout<<endl;

}

int main()

{

countd=10000;

goujian();

jagoubi();

gauss();

sor();

system("pause");

return 0;

}

# 体会与问题

1. 当lamuda值为1的时候，将迭代次数增加到100000，三个方法都可以得到最终的完美结果，随后逐渐减小迭代次数。将迭代次数减少到10000的时候，雅阁比方法不能得到最终的结果了，可见相对于这种取值雅阁比方法的效果差于其他两个方法。当迭代次数减少到8000的时候，只有高斯赛德尔方法能够得到最优结果，这里只能说明在w在0.5的取值情况下，效果不如高斯赛德尔方法。
2. 当lamuda值为0.1的时候，三种方法的迭代效果都很好，当迭代次数降到5000的时候才开始出现细微的差别，仍然是高斯赛德尔的方法效果更好，但是在这种取值的情况下，雅阁比方法的效果好于SOR。随着lamuda进一步减小，迭代效果变得更好，当lamuda值为0.01时，当迭代次数降到了2000，才出现了一些偏差，高斯赛德尔方法仍然是效果最好，雅阁比方法次之，0.5的SOR方法最差。当lamuda值为0.001时仍然是这种变化规律。
3. 在本实验之前我一直认为高斯消元法是解决这种问题的最好方法，但是通过这次实验我认识到对于这种三对角的稀疏矩阵还是使用这种迭代法效果更好，使用高斯消元法O(n3)的复杂度将不利于大规模的计算，但是个人认为可以通过判断标记矩阵内的非零元素减小计算的复杂度，不过代码编写起来会比较困难。