# 实验四

1. **实验内容**

4.1、考虑10阶矩阵作为系数的方程组，其中，的元素，。取初始解,编写程序用与迭代法求解该方程组，将作为终止迭代的判据。

（1）分别用与迭代法求解，观察收敛情况；

（2）改变的值，试验迭代法的效果，考察解的准确度。

1. **解题思路**

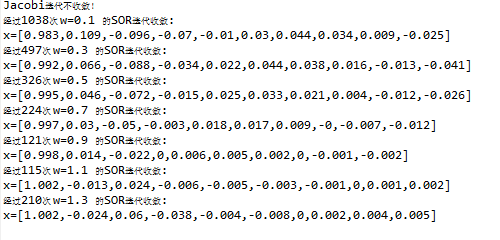
（注：程序均采用Java实现）

1、根据教材提供的Jacobi迭代算法和SOR迭代算法伪代码编程求解即可。

1. **算法设计**

|  |
| --- |
| **public** **class** Exp4  {  **public** **static** **double** norm(**double** x[] , **int** n) //计算向量的无穷范数  {  **double** max = Math.*abs*(x[0]);  **for** (**int** i=0;i<n;i++)  {  **if**(Math.*abs*(x[i]) > max)  {  max = Math.*abs*(x[i]);  }  }  **return** max;  }    **public** **static** **void** main(String[] args)  {  **final** **int** N= 10;    //初始化A矩阵  **double** A[][] = **new** **double** [N][N];  **for** (**int** i=0;i<N;i++)  {  **for** (**int** j=0;j<N;j++)  {  A[i][j] = 1.0/(i+j+1);  }  }    //初始化右端项b  **double** b[] = **new** **double** [N];  **for** (**int** i=0;i<N;i++)  {  b[i] = 1.0/(i+1);  }    //Jacobi迭代  **double** lastX[] = **new** **double** [N];  **double** nowX[] = **new** **double** [N];  **double** tmp[] = **new** **double** [N];  **int** counter = 0;  **while** (**true**)  {  counter++;  **for** (**int** i=0;i<N;i++)  lastX[i] = nowX[i];  **for** (**int** i=0;i<N;i++)  {  nowX[i] = b[i];  **for** (**int** j=0;j<i;j++)  {  nowX[i] -= A[i][j]\*lastX[j];  }  **for** (**int** j=i+1;j<N;j++)  {  nowX[i] -= A[i][j]\*lastX[j];  }  nowX[i] /= A[i][i];  }  **for** (**int** i=0;i<N;i++)  tmp[i] = nowX[i] - lastX[i];  **if**(*norm*(tmp , N) < 0.0001 || counter>10000)  **break**;  }    **if**(counter < 10000)  {  System.*out*.print("经过"+counter+"次Jacobi迭代：x=[");  **for** (**int** i=0;i<N-1;i++)  System.*out*.print(nowX[i]+",");  System.*out*.println(nowX[N-1]+"]");  }  **else**  {  System.*out*.println("Jacobi迭代不收敛！");  }        //SOR迭代  **for** (**double** w = 0.1;w<=1.3;w+=0.2)  {  counter = 0;  //初始化  **for**(**int** i=0;i<N;i++)  {  nowX[i] = lastX[i] = 0.0;  }      **while** (**true**)  {  counter++;  **for** (**int** i=0;i<N;i++)  lastX[i] = nowX[i];  **for** (**int** i=0;i<N;i++)  {  nowX[i] = b[i];  **for** (**int** j=0;j<i;j++)  {  nowX[i] -= A[i][j]\*nowX[j];  }  **for** (**int** j=i+1;j<N;j++)  {  nowX[i] -= A[i][j]\*nowX[j];  }  nowX[i] = nowX[i]/A[i][i]\*w;  nowX[i] = nowX[i]+(1-w)\*lastX[i];  }  **for** (**int** i=0;i<N;i++)  tmp[i] = nowX[i] - lastX[i];  **if**(*norm*(tmp , N) < 0.0001 || counter>10000)  **break**;  }  **if**(counter < 10000)  {  System.*out*.print("经过"+counter+"次"+" w="+w+" 的SOR迭代收敛:\nx=[");  **for** (**int** i=0;i<N-1;i++)  System.*out*.print(nowX[i]+",");  System.*out*.println(nowX[N-1]+"]");  }  **else**  {  System.*out*.println("w="+w+"的SOR迭代不收敛！");  }  }      }  } |

1. **实验结果和结论**



通过上述实验结果可知：越接近1，迭代收敛速度越快。