# 实验三

1. **实验内容**

3.6编程序生成矩阵，以及n维向量，其中为所有分量都是1的向量。用分解算法求解方程，得到近似解，计算残差和误差的-范数。

（1）设n=10，计算。

（2）在右端项上施加的扰动然后解方程组，观察残差和误差的变化情况。

（3）改变n的值为8和12，求解相应方程，观察的变化情况。通过这个实验说明了什么问题？

1. **解题思路**

（注：程序均采用java实现）

1. 根据矩阵的结构，生成n阶矩阵，继而得到向量；
2. 用分解算法求解方程，得到近似解，具体过程为：

首先对进行分解得到下三角矩阵，然后前代过程解下三角矩阵方程，得到向量，最后回代过程解上三角矩阵方程，得到近似解。

3、计算残差和误差的-范数。

1. **算法设计**

|  |
| --- |
| **public** **class** Exp3  {  **public** **static** **void** main(String[] args)  {  **final** **int** N = 10;    //计算hilbert矩阵  **double** [][]hilbert = **new** **double**[N][N];  **for** (**int** i=0;i<N;i++)  {  **for** (**int** j=0;j<N;j++)  {  hilbert[i][j] = 1.0/(i+j+1);  }  }    //计算向量b  **double** []b = **new** **double**[N];  **for** (**int** i=0;i<N;i++)  b[i] = 0.0;  **for** (**int** i=0;i<N;i++)  {  **for** (**int** j=0;j<N;j++)  {  b[i] += hilbert[i][j];  }  }    **double** a[][] = **new** **double** [N][N];  **for** (**int** i=0;i<N;i++)  {  **for** (**int** j=0;j<N;j++)  {  a[i][j] = hilbert[i][j];  }  }    //cholesky分解  **for** (**int** j=0;j<N;j++)  {  **for** (**int** k=0;k<j;k++)  {  a[j][j] = a[j][j] - a[j][k]\*a[j][k];  }  a[j][j] = Math.*sqrt*(a[j][j]);    **for**(**int** i=j+1;i<N;i++)  {  **for**(**int** k=0;k<j;k++)  {  a[i][j] = a[i][j] - a[i][k]\*a[j][k];  }  a[i][j] = a[i][j]/a[j][j];  }  }    **double** \_x[] = **new** **double** [N]; //第一次回带的中间结果  **double** x[] = **new** **double**[N];  **for** (**int** i=0;i<N;i++)  {  \_x[i] = b[i];  **for** (**int** j=0;j<i;j++)  \_x[i] -= (a[i][j]\*\_x[j]);  \_x[i] = \_x[i]/a[i][i];  }  **for** (**int** i=N-1;i>=0;i--)  {  x[i] = \_x[i];  **for** (**int** j=N-1;j>i;j--)  x[i] -= (a[j][i]\*x[j]);  x[i] = x[i]/a[i][i];  }      **double** r[] = **new** **double** [N]; //残差  **double** deltaX[] = **new** **double** [N]; //误差    **double** tmp[] = **new** **double**[N];  **for** (**int** i=0;i<N;i++)  {  deltaX[i] = x[i]-1;  **for** (**int** j=0;j<N;j++)  {  tmp[i] += hilbert[i][j]\*x[j];  }  r[i] = b[i] - tmp[i];  }    **double** max1 = Math.*abs*(deltaX[0]);  **double** max2 = Math.*abs*(r[0]);  **for** (**int** i=0;i<N;i++)  {  **if**(Math.*abs*(r[i]) > max1)  max1 = Math.*abs*(r[i]);  **if**(Math.*abs*(deltaX[i]) > max2)  max2 = Math.*abs*(deltaX[i]);  }  System.*out*.println("残差r的无穷范数为："+max1);  System.*out*.println("误差Δx为："+max2);  }    } |

1. **实验结果和结论**
2. n=10时，
3. 在右端项增加的扰动后：
4. n=8时，

n=12时，

进一步实验，整理实验结果如下表：

|  |  |  |  |
| --- | --- | --- | --- |
| N | 8 | 10 | 12 |
|  |  | 4.44585 | 0.335806 |
| 引入 | 0.0216222 | 0.700708 | 23.6202 |
|  |  | 2.08167 | 2.22045 |
| 引入 |  |  |  |

由上面结果可以看出：

1. 随着矩阵阶数n值变大，误差的无穷范数也变大，残差的无穷范数基本保持稳定。

若求问题的条件数，

我们知道矩阵是一种著名的病态矩阵，它的条件数，且随着n变大而变大。

由上表结果知，此问题的条件数cond，且随着n变大而变大。

因此说明此问题是敏感的。

1. 引入扰动后，误差的无穷范数明显增大，残差的无穷范数与扰动大小保持一致。