中北大学大数据学院实验报告

**课程名称： 数值分析 实验类型： 计算型**

**实验题目：**

班级：17070142学号：1707004716姓名：王浩 指导教师：韩慧妍 成绩：

|  |  |  |  |
| --- | --- | --- | --- |
| 实验地点 | 虚拟仿真实验室 | 实验时间 | 2019.12.4 |
| 1．实验目的  正确理解原始高斯消去法，并清楚其优缺点，正确理解列主元消去法，并且可以在程序中体现出其优势。 | | | |
| 2. 实验任务  用原始高斯消去法、列主元消去法分别求解方程组，并比较结果的精度 | | | |
| 3. 相关知识  注：与本实验相关的理论、知识点（原始高斯消去原理、列主元加入的原因及方法）  **原始高斯消去法**  基本思想：反复利用线性方程组初等变换中的一种变换，即用一个不为零的数成一个方程后加至另一个方程，使方程组变成同解的上三角形方程组，然后再自下而上对三角形方程组求解。高斯消去法可分为“消去”和“回代”两个过程。  图1 原始高斯消去法（三阶方程组为例）  将上述方法扩展到n方程联立方程组，当用k表示消元过程中的次序时，高斯消去法的计算步骤：   1. **消元过程：**   设对计算：  *;*  使用流程代码表示该过程，通过使用三个嵌套循环完成消元过程：   * **外层循环** 按照矩阵的主行依次从上到下进行； * **中层循环** 从当前主行开始，从上向下依次消去主元对应的变量； * **最内层循环** 处理列，计算一行内消去主元对应变量后面排列的其他变量的系数和等号右边的常数。  1. **回代过程、后向带入求解：**   ;  **列主元消去法**  列主消元的具体做法是：在进行第步消元时,对第列的以下选取绝对值最大的元素，然后通过行交换将其交换到的位置上。列选主元不影响求解的结果。即设主元在第个方程，即,若,将和方程互易位置，使新的成为主元，然后继续执行，这一步骤成为列选主元。 | | | |
| 4. 实验内容与实验结果（要求画出两种方法的流程图，并给出实验运行结果图，要求程序结果先输出个人姓名和学号，再输出实验结果，程序提交电子版即可）    图2 高斯消去法流程图 图3 列主元消去法流程图      图4 高斯消去法解方程组 图5 列主元消去法解方程组    图6 列主元消去法解手动输入方程组  **高斯、列主消元法python代码**  **import** numpy **as** np Mat = np.array([[0.9428, 0.3475, -0.8468, 0.4127],  [0.3475, 1.8423, 0.4759, 1.7321],  [-0.8468, 0.4759, 1.2147, -0.8621]])  *# 对上三角增广矩阵进行回代求解X* **def** Solution(Mat):  row = len(Mat) *# Mat矩阵的行数* column = len(Mat[0]) *# Mat矩阵的列数* X = np.zeros(row) *# 结果X向量* **for** i **in** range(row - 1, -1, -1):  b\_i = Mat[i][column - 1] *# bi的值* **for** j **in** range(i + 1, column - 1):  b\_i -= X[j] \* Mat[i][j] *# 减去已知解的占比* X[i] = b\_i / Mat[i][i] *# 除以所在行的主元素* **return** X, Mat  *# 高斯消去法求值* **def** Gaussian(Mat):  row = len(Mat) *# Mat矩阵的行数* column = len(Mat[0]) *# Mat矩阵的列数* **for** i **in** range(0,row - 1): *# 矩阵行下标i* **for** j **in** range(i + 1, row):  *# 计算乘数* m = Mat[j][i] / Mat[i][i]  *# 遍历第j行i到行尾的所有数* **for** k **in** range(i, column):  *# 每个数字进行相减* Mat[j][k] = Mat[j][k] - m \* Mat[i][k]  *# 输出每一步的数学意义* print(**"第{}行减去{}乘以第{}行得："**.format(j + 1, m, i + 1))  print(Mat) *# 输出该矩阵* print()  **return** Solution(Mat)  *# 列主消元法* **def** AllElimination(Mat):  row = len(Mat) *# Mat矩阵的行数* column = len(Mat[0]) *# Mat矩阵的列数  # 将矩阵化为上三角增广矩阵，并返回此矩阵* **for** i **in** range(row - 1): *# 矩阵行下标* maximum = 0  temp = i *# 保存主元行号  # 寻找列主元行* **for** j **in** range(i, row):  **if** Mat[j][i] > maximum:  maximum = Mat[j][i]  temp = j  *# 如果当年行不是列主元行，进行交换* **if** temp != i:  **for** j **in** range(i, column):  T = Mat[i][j]  Mat[i][j] = Mat[temp][j]  Mat[temp][j] = T  print(**"第{}行与第{}行进行交换："**.format(i + 1,temp + 1))  print(Mat)   **for** j **in** range(i + 1, row):  *# 计算乘数* m = Mat[j][i] / Mat[i][i]  *# 遍历第j行i到行尾的所有数* **for** k **in** range(i, column):  *# 每个数字进行相减* Mat[j][k] = Mat[j][k] - m \* Mat[i][k]  *# 输出每一步的数学意义* print(**"第{}行减去{}乘以第{}行得："**.format(j + 1, m, i + 1))  print(Mat) *# 输出该矩阵* print()   **return** Solution(Mat)  **if** \_\_name\_\_ == **'\_\_main\_\_'**:  print(**"1707004716 王浩"**)  print(**"1.使用默认数据 高斯消去法"**)  print(**"2.手动输入数据 高斯消去法"**)  print(**"3.使用默认数据 列主消元法"**)  print(**"4.手动输入数据 列主消元法"**)  boolean = int(input(**"请输入选择："**))  **if** boolean == 2 **or** boolean == 4:  n = int(input(**"请输入未知数个数："**))  Mat = np.zeros([n, n + 1])  **for** i **in** range(n):  **for** j **in** range(n + 1):  Mat[i][j] = float(input(**"请输入第{}行，第{}列数据："**.format(i + 1, j + 1)))  Mat = Mat.astype(np.float)  print(**"原矩阵为："**)  print(Mat)  **if** boolean == 1 **or** boolean == 2:  *# 高斯消去法* X, Mat = Gaussian(Mat)  **elif** boolean == 3 **or** boolean == 4:  *# 列主元消去法* X, Mat = AllElimination(Mat)  print(**"上三角增广矩阵为："**)  print(Mat)  print(**"列主元消去法求出的解为："**)  print(X) | | | |
| 5．总结  使用高斯消去法、列主消元法对求解的结果均为[-14.05184157 7.01437233 -13.2537575 ]，在使用列主消元法求解该方程组的过程中，列选主元时，搜索的最大的主元均为当前列主元，没有利用到列主消元的行互换，最终的结果也是一样的。当将方程换成时，使用原始高斯消去法求解结果为：[0,0.6667]。原因是因为使用了绝对值小的数字作为结果，使误差扩大，最终导致结果严重失真。而列主消元法选取绝对数大的数字做除数，避免了该问题，最终作为求解的结果为：[0.33333333 0.66666667]，与真实的方程结果：[0.3333,0.6667]非常接近。由此可见，精确度上来讲，列主消元法要高于高斯消去，但是由于找主元、换行浪费了时间，效率却不如高斯消去法。因此，在求解实际问题当中，要根据实际情况选择合适的方法。 | | | |