《数值分析》课程实验报告

实验名称 实验六 线性代数方程组的直接解法

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **班级** | 信息2202 | **姓名** | 徐梓乔 | **学号** | 220221100327 | **序号** |  |
| **教师** | 赵美玲 | **地点** | 数学实验中心 | | | **评分** |  |
| 1. 实验目的   1. 深入理解高斯消去法的基本原理，包括消元过程、矩阵的变换步骤以及算法的核心思想  2. 结合实际案例分析，掌握应对这些困难的改进措施，如列主元法的使用和优化算法的选择，在求逆和求行列式时提高问题求解的效率和准确性。   1. 实验过程和结果   1.解方程组    （1）顺序高斯消去法求解上述方程组，程序使用题目所给，验证所得到的解向量是否是原方程组的解，若不是原方程组的解，试分析原因，并证实你的分析的正确性！  （2）在顺序高斯消去法的程序基础上，将其改进为列主元高斯消去算法，调试通过并运行程序输出解向量，验证解向量是否是原方程组的解，写出你的结论和体会。  （3）利用MATLAB求方程组，计算出结果，对前面的方法进行验证。  解：  （1）利用题目所给的代码，运行“顺序高斯消去法”，得到：    验证：将解带入，得到三个方程的结果为1.1829992，2.1369993，3.03499933，应当可以当作是原方程的解，可以发现，其实对于这种各系数之间相差不大的方程，利用顺序高斯消去法的精度并不低。  （2）修改代码为“列主元高斯消去法”，运行得到：    很明显，与（1）中的解完全一致，正如（1）中所得出的结论一样，当系数相差不是特别大以及不是很小的时候，顺序法的误差会很小。  （3）写出MATLAB代码的脚本“gauss”，并运行得到：    可以看出得到的解是正确的。  三、思考题分析解答  1.（1）通过修改已经编译通过的顺序高斯消去算法和列主元高斯消去算法的程序计算：    写出相应的计算结果，并对结果进行分析！  （2）在已做的算例的基础上，思考为什么通常用列主元高斯消去法求解线性方程组？  （3）在解决某些问题时，方程组的系数不变，而右端的常数项不断发生变化，试问仍用列主元高斯消去算法求解有什么不足（从效率的角度考虑）？如何改进？并写出你的理由。  （4）参考以下矩阵LU分解的参考程序，试着编写解线性方程组的LU分解算法程序，并对以上线性方程组进行求解。（包括顺序高斯消去算法相对应的LU分解和列主元高斯消去算法所对应的LU分解）  （5）考虑用数值方法如何求方阵的行列式以及如何求可逆矩阵的逆矩阵？编程实现并用算例验证  解：  （1）分别运行文件夹练习思考中的代码“顺序高斯消去法”和“列主元高斯消去法”，得到如下结果：  顺序法：    带入原方程得到：59.16985；-39.90865；140.31784；-25.98399，可以看出结果发生了非常大的偏差，应当是由于系数过小造成的误差。  列主元：    带入原方程得到：59.1700128；46.77999401；0.999732；1.99995，显然列主元高斯消去法的精度非常好。  （2）在顺序高斯消去过程中，如果主元（对角线上的元素）接近零，计算中的除法可能会放大误差。列主元法通过选择列中绝对值最大的元素作为主元，显著降低了这种风险，上述（1）便是个很好的实例；计算机中使用浮点数表示实数，舍入误差在迭代过程中可能会积累。列主元法选择较大的主元，可以减少误差的累积，提高算法的数值稳定性。  （3）直接使用列主元高斯消去法，每次重新对整个系统进行消去，效率较低，每次重复消去中的元素是不必要的计算开销，改进的方法比如带列主元的LU分解，在完成对的LU分解后面对的改变无需再次进行有关系数矩阵的运输算，大大精简了此类情况的计算复杂程度。  （4）利用题设给出的代码，分别写出“LU顺序高斯消去法”和“LU列主元高斯消去法”，运行得到：    带入原方程得到：59.17；49.7498；0.892642；1.99282，可以看出经过LU分解后，结果以及比单纯的顺序高斯消法准确许多，但还是有误差。    带入原方程得：59.17；49.78；1.00007；2，结果相比于前面所有的都更加精确。  （5）只要经过LU分解后对L和U进行对角线相乘的操作即可得到行列式，而求逆则考虑使用LU分解求解方程的性质，将单位矩阵的每列作为方程的右端项解出数值并合并，就得到了逆矩阵。  写出并运行代码“LU求逆和行列式22222”得到：    进行简单的矩阵计算后可以看出结果很不错。  四、重点难点分析  1.本章的难点在于需要复杂的线性代数的计算以及将其使用C++语言程序化。  2.在撰写LU分解代码时，由于苯人C++的生疏，写数组时忘记了是从0开始计算，以至于一个小错误找了半个小时，真是令人感叹。 | | | | | | | |