

探讨用 C 语言实现求解行列式

屈晓

(华南农业大学 珠江学院, 广东 广州 510900)

摘要: 行列式在线性代数中是基础内容, 在对行列式进行求解时, 由于数值计算量比较大而且使用的方法量有一定的局限性, 所以在对行列式求解时, 数值的计算量比方法的寻找会更让人关注。所以该文提出了利用计算机来实现行列式的求解。利用计算机的计算速度优势, 可以对大量数据的行列式进行快速计算。该文研究的是用 C 语言实现行列式的降阶法求行列式的值。

关键词: C 语言; 行列式; 降阶法; 程序

中图分类号: TP271 文献标识码: A 文章编号: 1009-3044(2011)28-6907-02

行列式在线性代数中是基础的内容, 但由于其中的数值数据运算量大, 特别是高阶行列式, 显得比较复杂。并且在教学中只是挑选一些简单的例子进行说明、解释, 作业也比较少, 学生也没有足够的时间去深入学习, 致使学生对行列式的求解不够充分。虽然现在计算机已经普及, 但线性代数的教学并没有与算法语言的教学联系起来, 并没有用计算机来解决线性代数中的教学问题。对于行列式的计算而言, 很多学生只是掌握其中的一种或几种方法来解决行列式的计算, 这就花费了大量的时间在数值计算上, 并不在方法上。如果能用计算机来解决计算, 就可以大大的节约计算的时间。该文对行列式的求解, 利用 C 语言进行了一些算法上的探讨, 采用递归调用方式对行列式进行第一列展开进行求解, 从而能快速、通用的求出行列式的值。

1 行列式简述

令

$$D_n = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} = \sum_{j_1 j_2 \dots j_n} (-1)^{\tau(j_1 j_2 \dots j_n)} a_{1j_1} a_{2j_2} \dots a_{nj_n} \quad (1)$$

叫做 n 阶行列式。其中:

- 1) D_n 为一个数值, 其值为 $n!$ 项的代数和;
- 2) 每一项由 D_n 中不同行不同列的 n 个元素乘积组成;
- 3) 各项的符号由 n 级排列 $a_{1j_1} a_{2j_2} \dots a_{nj_n}$ 的奇偶性确定。

一般来说, 行列式的阶数越低越易计算, 因此很自然的考虑到能否把一个高阶行列式化为低阶行列式来计算。所以在此用余子式和代数余子式来解决行列式的计算, 利用代数余子式来对行列式按第一列进行展开, 并结合行列式的性质, 就可以简化行列式的计算。则行列式的计算可以转化为:

$$D_n = a_{1j} A_{1j} + a_{2j} A_{2j} + \dots + a_{nj} A_{nj} \quad (j=1, 2, \dots, n) \quad (2)$$

其中: A_{nj} 为代数余子式。

2 程序设计细节

2.1 数据结构

对于 $n \times n$ 的行列式来说, 存储采用二维数组是最容易想到的, 采用二维数组也是比较合理的。行列式的行数和列数是不确定的, 但是在 C 语言中, 定义数组时, 则必先确定数组的大小, 故可以采用一种办法解决, 就是在定义数组时确定数组的大小足够大。比如定义的数组为 `det[1000][1000]`。这种定义数组的方式存在一定的局限性: 1) 不能求解大于 1000×1000 的行列式; 2) 假如用该数组存储低阶行列式, 则浪费极大的内存空间。所以此文中不采用 `int det[1000][1000]` 的方式来定义数组。提出采用指针方式来定义数组, 则可以利用 `malloc()` 函数来实现动态的定义数组, 这样就可以根据用户的需求来分配内存空间。另外, 为了加快对数组的访问速度, 采用指针的方式进行访问数组, 程序的效率会得到大大的提高。故存储行列式采用的数据结构为:

```
int **det =& (int*) malloc(n * n * sizeof(int));
```

其中 n 的值为行列式的行数, 并且由用户输入决定。

2.2 实现计算方法

计算行列式的方法很多。比如: 定义法、三角形法、降阶法、递推法、数学归纳法以及加边升阶法。本文采用的方法为降阶法, 即按某一列进行展开, 将高阶的行列式化为低阶的行列式, 简化计算过程。由于在简化计算过程中, 每降低一阶时, 计算方式一样, 只是阶数减少一阶。故程序可以采用函数递归调用的方法来计算行列式, 大大降低了程序的复杂性, 使问题的解决变得更简单。程序

收稿日期: 2011-08-02

作者简介: 屈晓(1978-), 男, 湖南邵阳人, 讲师, 主要研究方向为计算机软件, 分布式计算与 Agent。

本栏目责任编辑: 谢媛媛

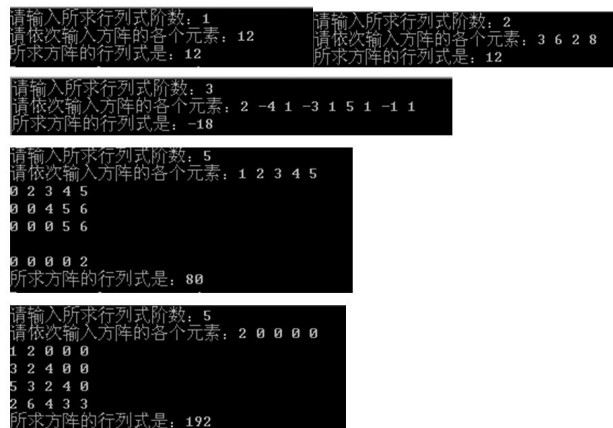
■■■■■■■■■■ 软件设计开发 ■■ 6907

关键代码段如下:

```
int solve_det(int n,int **a)
{
    ... ..
    if (n == 1)
        return a[0][0];
    for (i = 0;i < n;i++)
    {
        for (c = 0;c < n - 1;c++)
        {
            if(c < i)
                p = 0;
            else
                p = 1;
            for (j = 0;j < n-1;j++)
            {
                det[c][j] = a[c + p][j + 1];
            }
        }
        if (i % 2 == 0) q = 1;
        else q = -1;
        sum = sum + a[i][0] * q * solve_det(n - 1, det);
    }
    return sum;
    ... ..
}
```

3 测试结果

按照软件工程的测试要求,在 VC++6.0 环境下,对该程序进行了白盒测试和黑盒测试,发现程序能正常运行。部分测试数据和结果如下:



请输入所求行列式阶数: 1
请依次输入方阵的各个元素: 12
所求方阵的行列式是: 12

请输入所求行列式阶数: 2
请依次输入方阵的各个元素: 3 6 2 8
所求方阵的行列式是: 12

请输入所求行列式阶数: 3
请依次输入方阵的各个元素: 2 -4 1 -3 1 5 1 -1 1
所求方阵的行列式是: -18

请输入所求行列式阶数: 5
请依次输入方阵的各个元素: 1 2 3 4 5
0 2 3 4 5
0 0 4 5 6
0 0 0 5 6
0 0 0 0 2
所求方阵的行列式是: 80

请输入所求行列式阶数: 5
请依次输入方阵的各个元素: 2 0 0 0 0
1 2 0 0 0
3 2 4 0 0
5 3 2 4 0
2 6 4 3 3
所求方阵的行列式是: 192

由于该程序只是使用线性代数中的降阶法求行列式的值,而该算法经过大家的长期检验,是正确的,所以没有使用高阶行列式进行测试,也是没有必要的。

4 结束语

本文是使用 C 语言编写的对行列式进行第一列展开的方法进行计算,在程序中采用了二级指针来动态定义数组,并且采用了函数递归的方式对行列式按照列进行展开降阶的方法求解,这样使用在算法上大大简化,由于采用指针进行数据访问,所以程序运行的速度也得到了较大的提高。

参考文献:

- [1] 上海财经大学应用数学系.线性代数[M].上海:上海财经大学出版社,2004.
- [2] 谭浩强.C 程序设计[M].3 版.北京:清华大学出版社,2005.
- [3] 刘维富.C 语言程序设计一体化案例教程[M].北京:清华大学出版社,2009.
- [4] 杨起帆.C 语言程序设计教程[M].杭州:浙江大学出版社,2006.