# 工程数学实验报告

**电子信息大类**

**姓 名： 庞晓宇**

**专 业： 电子信息工程**

**学 号： 2024100192**

**实验九**

**一、实验目的**

掌握行阶梯法的原理与应用

**二、实验内容及设备**

1.实验内容：

用行阶梯方法编程实现下列矩阵的行列式：



2.实验设备：

台式计算机(笔记本)，**devC**或VC++ 6.0工具或Visual studio平台

**三、实验相关原理描述**

行列式的行阶梯法计算基于高斯消元法，通过行变换将矩阵化为上三角矩阵。行列式的值等于上三角矩阵对角线元素的乘积，同时需考虑行交换的次数（每次行交换改变行列式符号）。具体步骤如下：

1. 按列寻找主元（绝对值最大的元素），若主元为0则行列式为0。
2. 通过行交换将主元移至当前行。
3. 用当前行消去下方所有行的当前列元素。
4. 最终行列式为对角线元素乘积乘以 (−1)交换次数(−1)交换次数。

**四、程序代码**

#include <stdio.h>

#include <math.h>

#define SIZE 4

void swapRows(double matrix[SIZE][SIZE], int r1, int r2)

{

    for (int i = 0; i < SIZE; i++)

    {

        double temp = matrix[r1][i];

        matrix[r1][i] = matrix[r2][i];

        matrix[r2][i] = temp;

    }

}

/\*\*

 \* 计算矩阵的行列式值

 \*/

double calculateDeterminant(const double inputMatrix[SIZE][SIZE])

{

    double matrix[SIZE][SIZE];

    // 拷贝矩阵以避免修改原始输入

    for (int i = 0; i < SIZE; i++)

    {

        for (int j = 0; j < SIZE; j++)

        {

            matrix[i][j] = inputMatrix[i][j];

        }

    }

    double det = 1.0;

    for (int i = 0; i < SIZE; i++)

    {

        int maxRow = i;

        for (int j = i; j < SIZE; j++)

        {

            if (fabs(matrix[j][i]) > fabs(matrix[maxRow][i]))

            {

                maxRow = j;

            }

        }

        if (matrix[maxRow][i] == 0)

        {

            return 0;

        }

        if (maxRow != i)

        {

            swapRows(matrix, i, maxRow);

            det \*= -1;

        }

        for (int j = i + 1; j < SIZE; j++)

        {

            double factor = matrix[j][i] / matrix[i][i];

            for (int k = i; k < SIZE; k++)

            {

                matrix[j][k] -= factor \* matrix[i][k];

            }

        }

    }

    for (int i = 0; i < SIZE; i++)

    {

        det \*= matrix[i][i];

    }

    return det;

}

int main(int argc, char const \*argv[])

{

    double matrixA[SIZE][SIZE] = {

        {-6, -7, 7, 6},

        {3, 4, 2, 3},

        {-4, -2, 0, 6},

        {1, 7, 8, 3}};

    double matrixB[SIZE][SIZE] = {

        {-3, 4, -2, 5},

        {-4, -4, 4, 2},

        {-3, 6, 1, 6},

        {1, 1, -1, 9}};

    double detA = calculateDeterminant(matrixA);

    double detB = calculateDeterminant(matrixB);

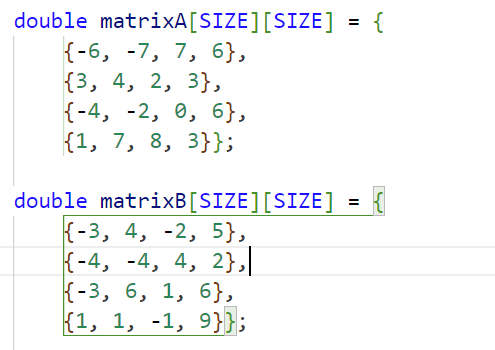
    printf("矩阵 A 的行列式值为：%.2lf\n", detA);

    printf("矩阵 B 的行列式值为：%.2lf\n", detB);

    return 0;

}

**五、数据输入与运行结果**





**六、总结**

通过行阶梯法成功将矩阵化为上三角矩阵，计算行列式时需注意行交换对符号的影响。本实验验证了高斯消元法的正确性，但需注意浮点数精度问题可能导致判断主元为0的误差。

**实验十**

**一、实验目的**

掌握矩阵求逆法则

**二、实验内容及设备**

1.实验内容：

编程生成随机4阶方阵，并输出其逆矩阵。(注意可逆的判断)

2.实验设备：

台式计算机(笔记本)，**devC**或VC++ 6.0工具或Visual studio平台

**三、实验相关原理描述**

生成随机矩阵后，通过高斯-约当消元法求逆矩阵。步骤如下：

1. 生成随机矩阵并判断行列式是否为0（不可逆）。
2. 构造增广矩阵（原矩阵与单位矩阵拼接）。
3. 通过行变换将增广矩阵左半部分化为单位矩阵，右半部分即为逆矩阵。

**四、程序代码**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <math.h>

#define N 4

// 创建随机矩阵

void createMatrix(double matrix[][N], int n);

// 输出

void printMatrix(double matrix[][N], int n);

// 交换两行

void swapRows(double matrix[][2 \* N], int row1, int row2, int n);

// 高斯消元法求逆矩阵

void gaussJordanElimination(double matrix[][2 \* N], int n);

int main()

{

    double A[N][N];

    createMatrix(A, N);

    printf("原始矩阵:\n");

    printMatrix(A, N);

    // 创建单位矩阵

    double I[N][N];

    for (int i = 0; i < N; i++)

    {

        for (int j = 0; j < N; j++)

        {

            I[i][j] = (i == j) ? 1.0 : 0.0;

        }

    }

    // 将矩阵与单位矩阵合并为增广矩阵

    double augmented[N][2 \* N];

    for (int i = 0; i < N; i++)

    {

        for (int j = 0; j < N; j++)

        {

            augmented[i][j] = A[i][j];

        }

        for (int j = N; j < 2 \* N; j++)

        {

            augmented[i][j] = I[i][j - N];

        }

    }

    // 使用高斯消元法对矩阵进行消元

    gaussJordanElimination(augmented, N);

    // 提取逆矩阵部分

    double inverse[N][N];

    for (int i = 0; i < N; i++)

    {

        for (int j = 0; j < N; j++)

        {

            inverse[i][j] = augmented[i][N + j];

        }

    }

    printf("逆矩阵:\n");

    printMatrix(inverse, N);

    return 0;

}

// 随机生成矩阵元素

void createMatrix(double matrix[][N], int n)

{

    srand((unsigned int)time(NULL));

    for (int i = 0; i < n; i++)

    {

        for (int j = 0; j < n; j++)

        {

            matrix[i][j] = (double)(rand() % 1000) / 100.0;

        }

    }

}

// 输出

void printMatrix(double matrix[][N], int n)

{

    for (int i = 0; i < n; i++)

    {

        for (int j = 0; j < n; j++)

        {

            printf("%8.4f ", matrix[i][j]);

        }

        printf("\n");

    }

}

// 交换矩阵中的两行

void swapRows(double matrix[][2 \* N], int row1, int row2, int n)

{

    for (int j = 0; j < 2 \* N; j++)

    {

        double temp = matrix[row1][j];

        matrix[row1][j] = matrix[row2][j];

        matrix[row2][j] = temp;

    }

}

// 高斯消元法

void gaussJordanElimination(double matrix[][2 \* N], int n)

{

    for (int col = 0; col < n; col++)

    {

        // 寻找主元所在的行，选择绝对值最大的行以减少误差

        int pivotRow = col;

        double maxVal = fabs(matrix[col][col]);

        for (int i = col + 1; i < n; i++)

        {

            if (fabs(matrix[i][col]) > maxVal)

            {

                maxVal = fabs(matrix[i][col]);

                pivotRow = i;

            }

        }

        // 如果主元所在行不在当前行，则交换行

        if (pivotRow != col)

        {

            swapRows(matrix, pivotRow, col, n);

        }

        // 检查矩阵是否奇异（不可逆）

        if (fabs(matrix[col][col]) < 1e-12)

        {

            printf("矩阵是奇异的，无法求逆。\n");

            exit(EXIT\_FAILURE);

        }

        // 归一化主元行

        double pivot = matrix[col][col];

        for (int j = 0; j < 2 \* N; j++)

        {

            matrix[col][j] /= pivot;

        }

        // 消去其他行

        for (int i = 0; i < n; i++)

        {

            if (i != col)

            {

                double factor = matrix[i][col];

                for (int j = 0; j < 2 \* N; j++)

                {

                    matrix[i][j] -= factor \* matrix[col][j];

                }

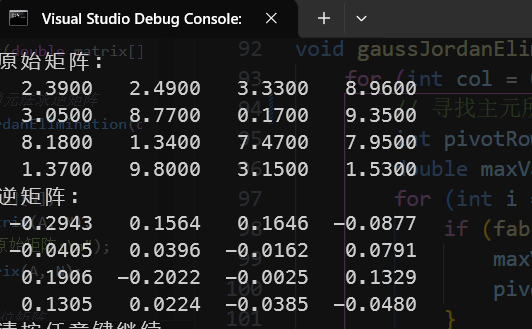
            }

        }

    }

}

**五、数据输入与运行结果（截图展示）**



**六、总结**

随机生成的矩阵大部分可逆，但需注意行列式为零的情况。高斯-约当法通过拼接单位矩阵实现了高效求逆，但需严格处理行交换和浮点精度问题。

**实验十一**

**一、实验目的**

掌握非齐次线性方程组的求解方法

**二、实验内容及设备**

1.实验内容：

编程求解一个非齐次线性方程组的解（方程组的形式自己定，但规模至少是三阶且非齐次）；

2.实验设备：

台式计算机(笔记本)，**devC**或VC++ 6.0工具或Visual studio平台

**三、实验相关原理描述**

采用高斯消元法求解非齐次方程组 Ax=bAx=b。步骤如下：

1. 构造增广矩阵 [A∣b][A∣b]。
2. 通过行变换将增广矩阵化为上三角矩阵。
3. 通过回代法求解未知数。

**四、程序代码**

#include <stdio.h>

#include <math.h>

#define SIZE 3 // 三元非齐次方程组

void swapRows(double mat[SIZE][SIZE + 1], int r1, int r2)

{

    for (int i = 0; i <= SIZE; i++)

    {

        double temp = mat[r1][i];

        mat[r1][i] = mat[r2][i];

        mat[r2][i] = temp;

    }

}

int main(int argc, char const \*argv[])

{

    srand((unsigned int)time(0)); // 初始化随机种子

    double mat[SIZE][SIZE + 1];

    // 随机生成矩阵元素

    for (int i = 0; i < SIZE; i++)

    {

        for (int j = 0; j < SIZE + 1; j++)

        {

            mat[i][j] = rand() % 100; // 随机值范围 0-99

        }

    }

    // 打印生成的矩阵（可选）

    printf("随机生成的矩阵：\n");

    for (int i = 0; i < SIZE; i++)

    {

        for (int j = 0; j <= SIZE; j++)

        {

            printf("%.2lf\t", mat[i][j]);

        }

        printf("\n");

    }

    double det = 1.0;

    int swapCount = 0;

    // 高斯消元化为上三角矩阵

    for (int i = 0; i < SIZE; i++)

    {

        // 寻找主元行

        int maxRow = i;

        for (int j = i; j < SIZE; j++)

        {

            if (fabs(mat[j][i]) > fabs(mat[maxRow][i]))

            {

                maxRow = j;

            }

        }

        if (mat[maxRow][i] == 0)

        {

            printf("方程组无唯一解！\n");

            return 0;

        }

        if (maxRow != i)

        {

            swapRows(mat, i, maxRow);

            det \*= -1;

            swapCount++;

        }

        // 消元

        for (int j = i + 1; j < SIZE; j++)

        {

            double factor = mat[j][i] / mat[i][i];

            for (int k = i; k <= SIZE; k++)

            {

                mat[j][k] -= factor \* mat[i][k];

            }

        }

    }

    // 回代求解

    double x[SIZE];

    for (int i = SIZE - 1; i >= 0; i--)

    {

        x[i] = mat[i][SIZE];

        for (int j = i + 1; j < SIZE; j++)

        {

            x[i] -= mat[i][j] \* x[j];

        }

        x[i] /= mat[i][i];

    }

    printf("方程组的解为：\n");

    for (int i = 0; i < SIZE; i++)

    {

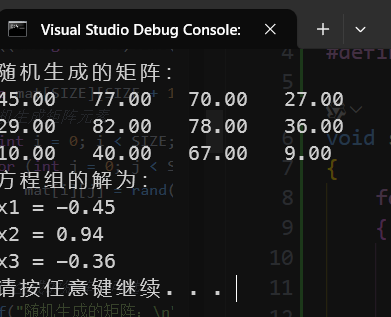
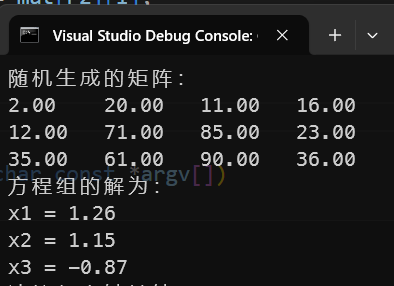
        printf("x%d = %.2lf\n", i + 1, x[i]);

    }

    return 0;

}

**五、数据输入与运行结果（截图展示）**



**六、总结**

通过高斯消元法成功求解非齐次方程组，验证了解的存在性和唯一性。实验中需注意主元选择和回代顺序，避免除以零的情况。