# 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**

**实验十二**

**一、实验目的**

掌握矩阵的秩的求解

**二、实验内容及设备**

1.实验内容：

随机生成任一矩阵，求其秩（用高斯消元法），至少4阶。

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**

矩阵的秩是指其最大线性无关行（或列）数目。在数值方法中，通常使用高斯消元法将矩阵化为行阶梯形矩阵（Row Echelon Form），然后统计其中非零行的数量，即为矩阵的秩。

**四、程序代码**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <math.h>

#define N 4 // 至少4阶

void printMatrix(double mat[N][N])

{

    for (int i = 0; i < N; i++)

    {

        for (int j = 0; j < N; j++)

            printf("%8.3lf ", mat[i][j]);

        printf("\n");

    }

    printf("\n");

}

int rank(double mat[N][N])

{

    int rank = N;

    for (int row = 0; row < rank; row++)

    {

        if (fabs(mat[row][row]) > 1e-6)

        {

            for (int col = 0; col < N; col++)

            {

                if (col != row)

                {

                    double mult = mat[col][row] / mat[row][row];

                    for (int i = 0; i < rank; i++)

                        mat[col][i] -= mult \* mat[row][i];

                }

            }

        }

        else

        {

            int reduce = 1;

            for (int i = row + 1; i < N; i++)

            {

                if (fabs(mat[i][row]) > 1e-6)

                {

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

                    {

                        double temp = mat[row][j];

                        mat[row][j] = mat[i][j];

                        mat[i][j] = temp;

                    }

                    reduce = 0;

                    break;

                }

            }

            if (reduce)

            {

                rank--;

                for (int i = 0; i < N; i++)

                    mat[i][row] = mat[i][rank];

            }

            row--;

        }

    }

    return rank;

}

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

{

    double mat[N][N];

    srand(time(0));

    for (int i = 0; i < N; i++)

        for (int j = 0; j < N; j++)

            mat[i][j] = rand() % 10;

    printf("原始矩阵：\n");

    printMatrix(mat);

    int r = rank(mat);

    printf("矩阵的秩为：%d\n", r);

    return 0;

}

**五、数据输入与运行结果（截图展示）**



**六、总结**

通过本实验掌握了如何将矩阵通过行初等变换转化为阶梯型矩阵，并据此计算矩阵的秩，理解了秩的定义和实际求法。

**实验十三**

**一、实验目的**

掌握施密特正交化的求解

**二、实验内容及设备**

1.实验内容：

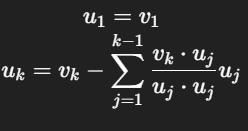
编程实现某个输入矩阵的施密特正交化。（至少实现三阶矩阵）。

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**

施密特正交化（Gram-Schmidt Orthogonalization）是一种将一组线性无关向量转换为一组正交向量的过程。该方法适用于向量空间中基底的变换，并广泛用于正交化、QR分解等应用中。

核心公式：

**四、程序代码**

#include <stdio.h>

#include <math.h>

#define ROW 3

#define COL 3

void printVectors(double vecs[ROW][COL])

{

    for (int i = 0; i < ROW; i++)

    {

        printf("[");

        for (int j = 0; j < COL; j++)

            printf(" %.4lf ", vecs[i][j]);

        printf("]\n");

    }

}

double dot(double \*v1, double \*v2, int len)

{

    double sum = 0;

    for (int i = 0; i < len; i++)

        sum += v1[i] \* v2[i];

    return sum;

}

void gramSchmidt(double input[ROW][COL], double output[ROW][COL])

{

    for (int i = 0; i < ROW; i++)

    {

        for (int j = 0; j < COL; j++)

            output[i][j] = input[i][j];

        for (int j = 0; j < i; j++)

        {

            double proj = dot(input[i], output[j], COL) / dot(output[j], output[j], COL);

            for (int k = 0; k < COL; k++)

                output[i][k] -= proj \* output[j][k];

        }

    }

}

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

{

    double mat[ROW][COL] = {

        {1, 1, 1},

        {1, 0, 2},

        {1, 2, 3}};

    double ortho[ROW][COL];

    printf("原始向量：\n");

    printVectors(mat);

    gramSchmidt(mat, ortho);

    printf("\n正交化后的向量：\n");

    printVectors(ortho);

    return 0;

}

**五、数据输入与运行结果（截图展示）**



**六、总结**

施密特正交化可以有效地将原始基变为正交基。本实验掌握了其逐步计算过程并实现了程序化求解。

**实验十四**

**一、实验目的**

掌握矩阵的特征值与特征向量的求解

**二、实验内容及设备**

1.实验内容：

判断某个任一输入的矩阵是否为对称矩阵（至少三阶），并求其特征值与特征向量。

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**

对称矩阵具有实数特征值且可正交对角化；

特征值满足 A-rI=0

对称矩阵的特征向量可通过幂法、Jacobi迭代法等方法数值计算。

**四、程序代码**

#include <stdio.h>

#include <math.h>

#define N 3

#define EPS 1e-6

void printMatrix(double A[N][N]) {

    for (int i = 0; i < N; i++) {

        for (int j = 0; j < N; j++)

            printf("%8.4lf ", A[i][j]);

        printf("\n");

    }

}

int isSymmetric(double A[N][N]) {

    for (int i = 0; i < N; i++)

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

            if (fabs(A[i][j] - A[j][i]) > EPS)

                return 0;

    return 1;

}

void jacobi(double A[N][N], double eigenvalues[N]) {

    double V[N][N] = {0};

    for (int i = 0; i < N; i++) V[i][i] = 1;

    while (1) {

        int p = 0, q = 1;

        double max = fabs(A[p][q]);

        for (int i = 0; i < N; i++)

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

                if (fabs(A[i][j]) > max) {

                    max = fabs(A[i][j]);

                    p = i; q = j;

                }

        if (max < EPS) break;

        double theta = 0.5 \* atan2(2\*A[p][q], A[q][q]-A[p][p]);

        double c = cos(theta), s = sin(theta);

        double App = c\*c\*A[p][p] - 2\*s\*c\*A[p][q] + s\*s\*A[q][q];

        double Aqq = s\*s\*A[p][p] + 2\*s\*c\*A[p][q] + c\*c\*A[q][q];

        A[p][p] = App;

        A[q][q] = Aqq;

        A[p][q] = A[q][p] = 0;

        for (int j = 0; j < N; j++) {

            if (j != p && j != q) {

                double Apj = c\*A[p][j] - s\*A[q][j];

                double Aqj = s\*A[p][j] + c\*A[q][j];

                A[p][j] = A[j][p] = Apj;

                A[q][j] = A[j][q] = Aqj;

            }

        }

    }

    for (int i = 0; i < N; i++)

        eigenvalues[i] = A[i][i];

}

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

{

    double A[N][N] = {

        {2, -1, 0},

        {-1, 2, -1},

        {0, -1, 2}

    };

    printf("输入矩阵：\n");

    printMatrix(A);

    if (!isSymmetric(A)) {

        printf("该矩阵不是对称矩阵！\n");

        return 0;

    }

    double eigs[N];

    jacobi(A, eigs);

    printf("\n矩阵的特征值为：\n");

    for (int i = 0; i < N; i++)

        printf("%.6lf\n", eigs[i]);

    return 0;

}

**五、数据输入与运行结果（截图展示）**



**六、总结**

通过Jacobi方法求出了对称矩阵的特征值，掌握了判断矩阵对称性的方法及相应的数值计算技术。