//矩阵转置

public double[,] tranMatrix(double[,] a)

{

int m, n;

m = a.GetLength(0); //获取二维数组即矩阵中第一维的长度，即行数

n = a.GetLength(1); //获取矩阵中第二维的长度，即列数

double[,] b = new double[n, m]; //定义新的矩阵为n行m列，用于存放转置后矩阵

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

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

b[i, j] = a[j, i];

return b;

}

//矩阵相乘

public double[,] MultiplyMatrix(double[,] a, double[,] b)

{

int m, n, M, N;

m = a.GetLength(0);//获取a,b两个矩阵的行列数

n = a.GetLength(1);

M = b.GetLength(0);

N = b.GetLength(1);

double[,] c = new double[m, N];//定义新的矩阵为m行N列，用于存放相乘后矩阵

if (n != M)

MessageBox.Show("输入的两个矩阵不能相乘！");//检查矩阵a的列数是否等于b的行数

else

{

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

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

{

double sum = 0;

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

sum += a[i, k] \* b[k, j]; //将矩阵a的第i行与b的第j列对应相乘再相加

c[i, j] = sum;

}

}

return c;

}

//矩阵求逆--首先计算矩阵对应行列式的值

public static double MatrixValue(double[,] MatrixList) //首先需要计算行列式的值|A|， 如果为0 说明不可逆

//计算行列式的方法

// a1 a2 a3

// b1 b2 b3

// c1 c2 c3

// 结果为a1·b2·c3+b1·c2·a3+c1·a2·b3-a3·b2·c1-b3·c2·a1-c3·a2·b1(注意对角线就容易记住了)

{

int Level = MatrixList.GetLength(0); //获取矩阵维数

double[,] dMatrix = new double[Level, Level]; //定义二维数组,行列数相同

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

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

dMatrix[i, j] = MatrixList[i, j]; //将参数的值,赋值给定义的数组

double c, x;

int k = 1;

for (int i = 0, j = 0; i < Level && j < Level; i++, j++)

{

if (dMatrix[i, j] == 0) //判断对角线上的数据是否为0（从i=j=0即第一个数据开始）

{

int m = i;

for (; dMatrix[m, j] == 0; m++) ; //如果对角线上数据为0,从该数据开始依次往后判断该列是否为0

if (m == Level) //当该列从对角线数据开始数据都为0 的时候返回0

return 0;

else

{

// Row change between i-row and m-row从对角线数据开始交换第i行和第m行数据，因为对角线之前都是0

for (int n = j; n < Level; n++)

{

c = dMatrix[i, n];

dMatrix[i, n] = dMatrix[m, n];

dMatrix[m, n] = c;

}

// Change value pre-value 交换两行后行列式变号

k \*= (-1);

}

}

// Set 0 to the current column in the rows after current row 化成下三角

for (int s = Level - 1; s > i; s--)

{

x = dMatrix[s, j];

for (int t = j; t < Level; t++)

dMatrix[s, t] -= dMatrix[i, t] \* (x / dMatrix[i, j]);

}

}

double sn = 1;

for (int i = 0; i < Level; i++) //化成下三角后，行列式值为对角线相乘

{

if (dMatrix[i, i] != 0)

sn \*= dMatrix[i, i];

else

return 0;

}

return k \* sn;

}

//矩阵求逆—开始求逆

public static double[,] ReverseMatrix(double[,] dMatrix) //开始求逆

{

int Level = dMatrix.GetLength(0); //获取矩阵维数

double dMatrixValue = MatrixValue(dMatrix); //求取矩阵的行列式值

if (dMatrixValue == 0) return null; //A为该矩阵若|A| =0 则该矩阵不可逆返回空

double[,] dReverseMatrix = new double[Level, 2 \* Level]; //列为行的2倍，用于存放单位阵I

double x, c;

// Init Reverse matrix

for (int i = 0; i < Level; i++) //创建一个矩阵(A|I)以对其进行初等变换求得其矩阵的逆

{

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

{

if (j < Level)

dReverseMatrix[i, j] = dMatrix[i, j]; //该(A|I)矩阵前Level列为矩阵A 后面为数据全部为0

else

dReverseMatrix[i, j] = 0;

}

dReverseMatrix[i, Level + i] = 1;

//将Level+1行开始的Level阶矩阵装换为单位矩阵(起初的时候该矩阵都为0 现在在把对角线位置装换为1 )

//参考http://www.shuxuecheng.com/gaosuzk/content/lljx/wzja/12/12-6.htm

}

for (int i = 0, j = 0; i < Level && j < Level; i++, j++)

{

if (dReverseMatrix[i, j] == 0) //判断一行对角线是否为0

{

int m = i;

for (; dMatrix[m, j] == 0; m++) ;

if (m == Level)

return null; //某行对角线为0的时候判断该行该数据所在的列在该数据后是否为0 都为0 的话不可逆返回空值

else

{

// Add i-row with m-row

for (int n = j; n < 2 \* Level; n++) //如果对角线为0 则该i行加上m行 m行为(初等变换要求对角线为1,0-->1先加上某行,下面在变1)

dReverseMatrix[i, n] += dReverseMatrix[m, n];

}

}

// 此时数据:第二行加上第一行为第一行的数据

// 1 1 3 1 1 0

// 1 0 1 0 1 0

// 4 2 1 0 0 1

//

// Format the i-row with "1" start

x = dReverseMatrix[i, j];

if (x != 1) //如果对角线元素不为1 执行以下

{

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

if (dReverseMatrix[i, n] != 0)

dReverseMatrix[i, n] /= x; //相除使i行第一个数字为1

}

// Set 0 to the current column in the rows after current row

for (int s = Level - 1; s > i; s--) //该对角线数据为1 时,这一列其他数据要转换为0

{

x = dReverseMatrix[s, j];

// 第一次时

// 1 1 3 1 1 0

// 1 0 1 0 1 0

// 4(x) 2 1 0 0 1

//

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

dReverseMatrix[s, t] -= (dReverseMatrix[i, t] \* x);

//第一个轮回用第一行\*4 减去第三行为第三行的数据依次类推

// 1 1 3 1 1 0

// 1 0 1 0 1 0

// 0(x) -2 -11 -4 -4 1

}

}

// Format the first matrix into unit-matrix

for (int i = Level - 2; i >= 0; i--)

//处理第一行二列的数据思路如上就是把除了对角线外的元素转换为0

{

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

if (dReverseMatrix[i, j] != 0)

{

c = dReverseMatrix[i, j];

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

dReverseMatrix[i, n] -= (c \* dReverseMatrix[j, n]);

}

}

double[,] dReturn = new double[Level, Level];

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

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

dReturn[i, j] = dReverseMatrix[i, j + Level];

//就是把Level阶的矩阵提取出来(减去原先为单位矩阵的部分)

return dReturn;

}