矩阵运算代码

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

{

int m, n;

m = a.GetLength(0);

n = a.GetLength(1);

double[,] b = new double[n, m];

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

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

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

return b;

}

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

{

int m, n, M, N;

m = a.GetLength(0);

n = a.GetLength(1);

M = b.GetLength(0);

N = b.GetLength(1);

double[,] c = new double[m, N];

if (n != M)

MessageBox.Show("输入的两个矩阵不能相乘！");

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];

c[i, j] = sum;

}

}

return c;

}

public static double MatrixValue(double[,] MatrixList)

{

int Level = MatrixList.GetLength(0);

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

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

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

dMatrix[i, j] = MatrixList[i, j];

double c, x;

int k = 1;

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

{

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

{

int m = i;

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

if (m == Level)

return 0;

else

{

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

{

c = dMatrix[i, n];

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

dMatrix[m, n] = c;

}

k \*= (-1);

}

}

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++)

{

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

sn \*= dMatrix[i, i];

else

return 0;

}

return k \* sn;

}

public static double[,] ReverseMatrix(double[,] dMatrix) {

int Level = dMatrix.GetLength(0);

double dMatrixValue = MatrixValue(dMatrix);

if (dMatrixValue == 0) return null;

double[,] dReverseMatrix = new double[Level, 2 \* Level];

double x, c;

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

{

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

{

if (j < Level)

dReverseMatrix[i, j] = dMatrix[i, j];

else

dReverseMatrix[i, j] = 0;

}

dReverseMatrix[i, Level + i] = 1;

}

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

{

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

{

int m = i;

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

if (m == Level)

return null;

else

{

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

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

}

}

x = dReverseMatrix[i, j];

if (x != 1)

{

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

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

dReverseMatrix[i, n] /= x;

}

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

{

x = dReverseMatrix[s, j];

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

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

}

}

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

{

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];

return dReturn;

}