// This does not take into account the negative and positive sign

float laplaceExtension(const float \*matrix, const int dimension)

{

if (dimension == 2)

{

return (matrix[0] \* matrix[3]) - (matrix[1] \* matrix[2]);

}

float det = 0;

float \* mat = new float[(dimension - 1) \* (dimension - 1)];

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

{

int currentIndex = 0;

for (int y = 1; y < dimension; y++)

{

for (int x = 0; x < dimension; x++)

{

if (x == i)

continue;

mat[currentIndex++] = matrix[x + y \* dimension];

}

}

det += matrix[i] \* laplaceExtension(mat, dimension-1);

}

delete[] mat;

return det;

//float det = 0;

//dimension -= 1;

//float \* mat = new float[(dimension) \* (dimension)];

//for (int i = 0; i < 4; i++)

//{

// int sX = 0;

// int sY = 0;

// int mX = 0;

// int mY = 0;

// switch (i)

// {

// case 0:

// {

// mX = 1;

// mY = 1;

// break;

// }

// case 1:

// {

// sX = dimension;

// mY = 1;

// break;

// }

// case 2:

// {

// sY = dimension;

// mX = 1;

// break;

// }

// case 3:

// {

// sX = dimension;

// sY = dimension;

// break;

// }

// }

// for (int x = 0; x < dimension; x++)

// {

// for (int y = 0; y < dimension; y++)

// {

// mat[x + y \* dimension] = matrix[(x + mX) + (y + mY) \* (dimension + 1)];

// }

// }

//

// float val = laplaceExtension(mat, dimension);

// if ((sX + sY) % 2 != 0)

// val = -val;

// det += matrix[sX + sY \*(dimension +1)] \* val;

//}

//delete[] mat;

//return det;