public void Solve()

{

double max;

//вектор, в который записывается текущее приближение

curr\_vector = new double[X.Values.Count, 1];

//вектор, в котором хранится предыдущее приближение

prev\_vector = new double[X.Values.Count, 1];

//Матрица значений, получаемая из начальных выражений и подставленных в них приближений

curr\_func\_value = new double[Equations.Functions.GetLength(1), 1];

//Создание временного словаря и присвоение ему начальных значений

var temp = X;

do

{

//Увеличение счетчика итераций

Counter++;

//Составление численного Якобиана из обновленных значений перменных

numJuc = JacobianToDoubleFormat(temp);

//Нахождение обратной матрицы Якобиана

revJacobian = FindRevJacobian(numJuc);

//Обновление матрицы приближенных значений предыдущего шага

for (int i = 0; i < temp.Count; i++)

{

var elem = temp.ElementAt(i);

prev\_vector[i, 0] = double.Parse(elem.Value.RealValue.ToString().Replace(".", ","));

}

//Расчет значений для матрицы функций

for (int i = 0; i < Equations.Functions.GetLength(0); i++)

{

for (int j = 0; j < Equations.Functions.GetLength(1); j++)

{

curr\_func\_value[i, 0] += double.Parse(Equations.Functions[i, j].Evaluate(temp).RealValue.ToString().Replace(".", ","));

}

}

//Расчет дельта X

var delta\_x = Matrix.MatrixMultiply(revJacobian, curr\_func\_value);

//Получение нового приближения

curr\_vector = Matrix.MatrixSubtraction(prev\_vector, delta\_x);

max = Math.Abs(delta\_x[0, 0]);

temp = new Dictionary<string, FloatingPoint>();

//Вычисление максимального значения в массиве delta\_x и обновление словаря temp

for (int i = 0; i < X.Count; i++)

{

if (max < Math.Abs(delta\_x[i, 0]))

max = Math.Abs(delta\_x[i, 0]);

var elem = X.ElementAt(i);

temp.Add(elem.Key, Expr.Parse(curr\_vector[i, 0].ToString().Replace(",", ".")).RealNumberValue);

}

} while (max > Eps && Counter != maxCounter);

//Вывод количества ит

Console.WriteLine($"Итоговое количество итераций: {Counter}");

Console.WriteLine("Ответ: ");

Matrix.PrintMatrix(curr\_vector);

Console.WriteLine("\n");

}