МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ, МОЛОДЕЖИ И СПОРТА УКРАИНЫ

ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ РАДИОЭЛЕКТРОНИКИ

Кафедра ИИ

Отчёт

Дисциплина: «Технология распределённых систем и параллельных исчислений»

Тема: «Создание многопоточных приложений под платформу .NET»

Проверил:

Выполнил:

Губин В.А.

Неблиенко М.

2014

ЛАБОРАТОРНАЯ РАБОТА №1

НАЗВАНИЕ РАБОТЫ: Создание многопоточных приложений под платформу .NET.

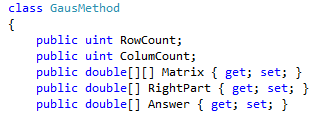
ЦЕЛЬ РАБОТЫ: Изучение особенностей использования делегатов и классов библиотеки System.Threading для создания многопоточных приложений под платформу .NET. Изучение особенностей распараллеливания математических алгоритмов.

ЗАДАНИЕ 5\*\*\*. Решение системы линейных уравнений методом Гаусса

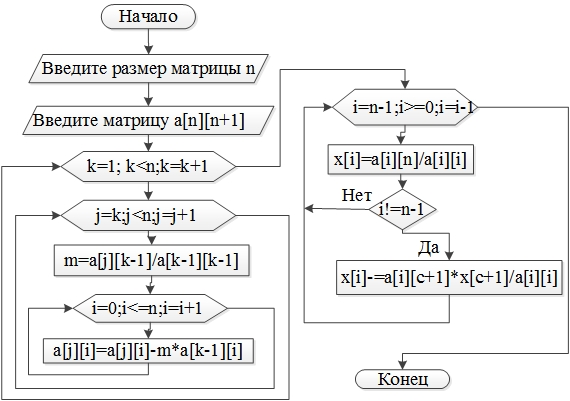
Данные класса: матрица N\*N и правая часть размером N.

Функции класса: решение линейного уравнения методом Гаусса.

ИНТЕРФЕЙС РАЗРАБОТАННОГО КЛАССА:



АЛГОРИТМ РАБОТЫ РАЗРАБОТАННОГО КЛАССА:



РЕАЛИЗАЦИЯ РАЗРАБОТАННОГО КЛАССА:

using System;

using System.Collections.Generic;

using System.Text;

using System.Threading;

namespace PPP1

{

class GausMethod

{

private uint RowCount;

private uint ColumCount;

public double[][] Matrix { get; set; }

public double[] RightPart { get; set; }

public double[] Answer { get; set; }

private delegate void dSearchForAnswer(double[] Answer, double[] RightPart, uint RowCount, double[][] Matrix, int i);

private dSearchForAnswer searcher = SearchForAnswer;

public GausMethod(uint Row, uint Column)

{

RightPart = new double[Row];

Answer = new double[Row];

Matrix = new double[Row][];

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

{

Matrix[i] = new double[Column];

}

RowCount = Row;

ColumCount = Column;

//обнулим массив

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

{

Answer[i] = 0;

RightPart[i] = 0;

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

{

Matrix[i][j] = 0;

}

}

}

public void SolveMatrix()

{

if (RowCount != ColumCount)

{

throw new ArithmeticException("Нет решений");

}

for (int i = 0; i < RowCount - 1; i++)

{

Minimize(i);

}

IAsyncResult[] process = new IAsyncResult[RowCount];

//ищем решение

for (int i = (int)(RowCount - 1); i >= 0; i--)

{

process[i] = searcher.BeginInvoke(Answer, RightPart, RowCount, Matrix, i, null, null);

}

s: while (true)

{

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

if (!process[j].IsCompleted) goto s;

break;

}

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

{

searcher.EndInvoke(process[j]);

}

}

private void Minimize(int i)

{

SortRows(i);

AutoResetEvent[] events = new AutoResetEvent[RowCount - 1 - i];

MultyThreadData[] threads = new MultyThreadData[RowCount - 1 - i];

for (int j = 0; j < (RowCount - i - 1); j++)

{

events[j] = new AutoResetEvent(false);

threads[j] = new MultyThreadData(events[j], Matrix, RightPart, i, j + 1 + i);

ThreadPool.QueueUserWorkItem(this.Handle, threads[j]);

}

WaitHandle.WaitAll(events);

}

private void Handle(Object obj)

{

MultyThreadData thread = (MultyThreadData)obj;

double[][] Matrix = thread.Matrix;

double[] RightPart = thread.RightPart;

int i = thread.i;

int j = thread.j;

if (Matrix[i][i] != 0) //если главный элемент не 0, то производим вычисления

{

double MultElement = Matrix[j][i] / Matrix[i][i];

for (int k = i; k < ColumCount; k++)

{

Matrix[j][k] -= Matrix[i][k] \* MultElement;

}

RightPart[j] -= RightPart[i] \* MultElement;

}

//для нулевого главного элемента просто пропускаем данный шаг

thread.e.Set();

}

private static void SearchForAnswer(double[] Answer, double[] RightPart, uint RowCount, double[][] Matrix, int i)

{

Answer[i] = RightPart[i];

for (int j = (int)(RowCount - 1); j > i; j--)

Answer[i] -= Matrix[i][j] \* Answer[j];

if (Matrix[i][i] == 0)

if (RightPart[i] == 0)

throw new ArithmeticException("Несколько решений");

else

throw new ArithmeticException("Нет решений");

Answer[i] /= Matrix[i][i];

}

private void SortRows(int SortIndex)

{

double MaxElement = Matrix[SortIndex][SortIndex];

int MaxElementIndex = SortIndex;

for (int i = SortIndex + 1; i < RowCount; i++)

{

if (Matrix[i][SortIndex] > MaxElement)

{

MaxElement = Matrix[i][SortIndex];

MaxElementIndex = i;

}

}

//теперь найден максимальный элемент ставим его на верхнее место

if (MaxElementIndex > SortIndex)//если это не первый элемент

{

double Temp;

Temp = RightPart[MaxElementIndex];

RightPart[MaxElementIndex] = RightPart[SortIndex];

RightPart[SortIndex] = Temp;

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

{

Temp = Matrix[MaxElementIndex][i];

Matrix[MaxElementIndex][i] = Matrix[SortIndex][i];

Matrix[SortIndex][i] = Temp;

}

}

}

private class MultyThreadData

{

public MultyThreadData(AutoResetEvent e, double[][] Matrix, double[] RightPart, int i, int j)

{

this.e = e;

this.i = i;

this.j = j;

this.Matrix = Matrix;

this.RightPart = RightPart;

}

public AutoResetEvent e { get; set; }

public double[][] Matrix { get; set; }

public double[] RightPart { get; set; }

public int i { get; set; }

public int j { get; set; }

}

}

class Program

{

static void Main(string[] args)

{

const int N = 3;

GausMethod Solution = new GausMethod(N, N);

DateTime start = System.DateTime.Now;

int counter = 1;

//заполняем правую часть

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

{

Solution.RightPart[i] = counter++;

}

counter = 1;

//заполняем матрицу

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

{

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

{

Solution.Matrix[i][j] = counter++;

}

}

//решаем матрицу

try

{

Solution.SolveMatrix();

//выводим ответ

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

{

Console.WriteLine(Solution.Answer[i]);

}

}

catch (ArithmeticException e)

{

Console.WriteLine(e.Message);

}

DateTime stop = System.DateTime.Now;

Console.WriteLine("Time elapsed for calculation : {0}", stop - start);

Console.ReadKey();

}

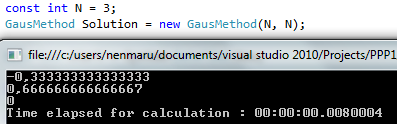
}

}

РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ:

Малые величины (N=3):

Многопоточное приложение:

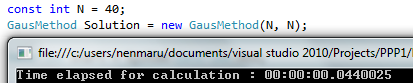


Однопоточное приложение:



Большие величины (N=40):

Многопоточное приложение:



Однопоточное приложение:



ВЫВОДЫ ПО РАБОТЕ: При небольших объёмах вычисляемых данных многопоточность может только повредить быстродействию. Но при больших объёмах данных она станет хорошим подспорьем для ускорения работоспособности приложения.