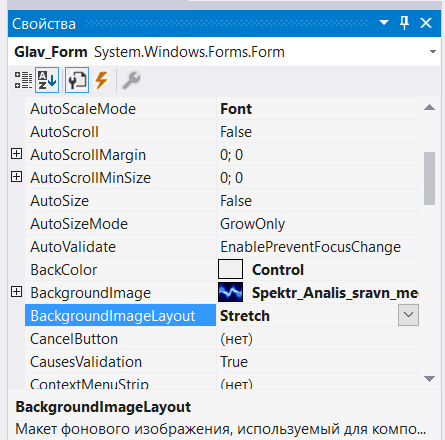
# Введение

Здесь я буду описывать процесс создания этой программы, чтобы в будущем не сталкиваться с такими проблемами, а точнее, чтобы потом мучительно не вспоминать «А как же это делается». Этот документ отнюдь не является учебником, и в нём могут быть пропущенным многие моменты, ибо они могли быть понятны и не вызывать никаких сложностей. Так же тут будут и эмоции и подобное, ибо это дневник, а не какое-то пособие.

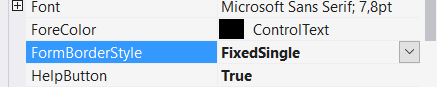
# Создание форм и связь между ними

21.06.2017

Я начал эту программу, первое что я сделал, это открыл её собственно, то есть создал пустой виндовсовский проект. И решил сделать нормальной хотя бы титульную страницу, в плане «красивой». Первое что сделал, это нажал на форму один раз и справа снизу отобразились её свойства, в принципе там нет ничего сложного, они объяснены на русском. Само собой я сразу поменял название формы. Затем добавил фоновую картинку, которую я, стоит заметить, довольно долго искал в интернете, пытаясь подобрать что-то более менее по теме, и чтобы радовало глаз. Вот это окошко со свойствами, тут я подмечу несколько фактов.



Выбранное свойство говорит о том, где будет расположено окно, когда запустится программа, у меня выбрано «по центру», может оно и не совсем так, но я это определил опытным путём.



Данное свойство говорит о том, можно ли будет растягивать форму, я это свойство убрал, мне оно показалось лишним и добавляющим проблемы, к тому же для этой формы это явно лишнее



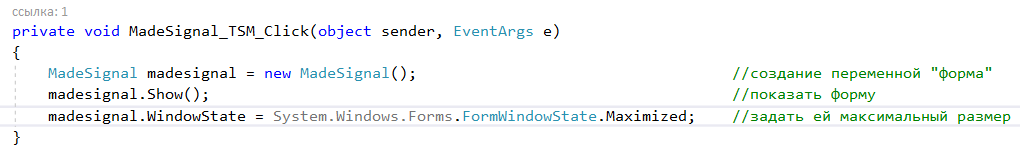
Что это? Да всё просто, это иконка, которая будет отображаться в панели задач, да и в левом верхнем углу. И вот это довольно много мне проблем доставило, ибо добавить-то не сложно, а вот как собственно создать файл с таким расширение? То есть я его создал, но как-то не так, он ругался. В общем решение такое – зайти в гугл, и найти онлайн конверт типов, вот и всё, само собой расширение .ico .

Размеры формы я задал по размерам картинки, ну а зачем её искажать? И само собой исправил текст, пусть он будет читабелен по русский (это текст в заголовке, в самом верху).

С этим вопросы оформления формы можно считать завершёнными (то есть чисто формы). Дальше я перешёл к добавлению менюшки, это в «панель элементов», слева выпадающие меню, и там «MenuStrip». Само собой её нужно перетащить на форму, затем сразу всё что положено переименовать. Ну а затем уже добавить интересующие пункты меню (тоже сразу их именую нужным образом). Сейчас всё это выглядит следующим образом



Теперь про переход между формами. Я сделал новую форму, и она открывается поверх существующей, да, возможно не лучшая идея, но она работает и хай так будет, пока что сойдёт. Для этого в вкладке меню с спектральным анализом, я добавил пункт «создание сигнала», и для него уже открыл всё что нужно. А именно, нужно кликнуть на этой вкладке, и дальше в свойствах, выбрать действие и там уже на «Click» нажать два раза и уже внутри этого прописывать следующий код (чтобы именно при нажатие открывалось окно).



MadeSignal madesignal = new MadeSignal();

madesignal.Show();

madesignal.WindowState = System.Windows.Forms.FormWindowState.Maximized;

Само собой я сначала создал эту форму с именем «MadeSignal». И оформил её по тем же принципам. Код комментировать не нужно, всё прописано и так.

# Работа с excel файлом

Вот тут у меня вчера начались проблемы, знатные проблемы, во-первых я как минимум не мог подключить нужны библиотеки… сейчас распишу путь, по которому это делается «Вид - Обозреватель решиний - Правой кнопкой: Ссылки Добавить ссылку. На вкладке .net добавляем сборку Microsoft.Office.Interop». По этому поводу я вчера очень долго тупил, и с трудом нашёл решение в интернете, благо, что вообще нашёл. Спасибо этим людям.

Затем уже подключается библиотека

using Excel = Microsoft.Office.Interop.Excel;

А вот со всем что идёт дальше… тут я ещё не разобрался, так что на этом остановлюсь и начну читать интернет, разбираться

22.06.2017

В итоге я создал этот класс, на данный момент он читает из экселя. Но это уже ОЧЕНЬ много, и сколько я на это времени потратил… это просто пиздец… Пытаясь разобраться, ибо пример с сайте он не работал, хотя и был верный, просто мне его пришлось переправить

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Microsoft.Office.Interop.Excel;

using System.Runtime;

namespace Spektr\_Analis\_sravn\_medods

{

// Этот класс содержит код Excel Interop нам нужно.

// Это могут быть разделены во многих местах, чтобы избежать дублирования.

class ExcelReaderInterop

{

// Сохранить объект Application можно использовать в функциях-членах.

Application \_excelApp;

// Инициализировать новый читатель Excel. Должны быть интегрированы

// с объектом интерфейса Excel.

public ExcelReaderInterop()

{

\_excelApp = new Application();

}

// Метод, который открывает книги Excel: C #

// Открыть путь к файлу, полученный в Excel. Затем откройте книгу

// в файле. Отправить книгу к следующей функции, внутреннее сканирование

// функция. Будет сгенерировано исключение, если файл не может быть найден или открыт.

public object[,] ExcelReadTableOneShets(string thisFileName, int numSheet)

{

try

{

// Этот беспорядок кода открывает книгу Excel. Я не знаю, что все

// эти аргументы делать, но они могут быть изменены, чтобы влиять на поведение.

Workbook workBook = \_excelApp.Workbooks.Open(thisFileName,

Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing);

object[,] valueTable = ExcelScanIntenalOneShet(workBook, numSheet);

// закрытие книги

Excelclose(workBook, thisFileName);

return valueTable;

}

catch

{

//

// Обработка исключений.

//

return null;

}

}

// Метод, который закрывает книги Excel: C #

// Открыть путь к файлу, полученный в Excel. Затем откройте книгу

// в файле. Отправить книгу к следующей функции, внутреннее сканирование

// функция. Будет сгенерировано исключение, если файл не может быть найден или открыт.

private void Excelclose(Workbook workBook, string thisFileName)

{

try

{

//

// Очистка.

//

workBook.Close(false, thisFileName, null);

System.Runtime.InteropServices.Marshal.ReleaseComObject(workBook);

//именно так прописывать, чтобы был полный путь. функция возвращает число оставшихся ссылок.

}

catch

{

//

// Обработка исключений.

//

}

}

// Метод, который получает лист1: C # (ОДИН ЛИСТ)

// Сканирование выбранной книги Excel и сохранять информацию в ячейках

// для этой книги в качестве объекта [,] массив. Затем, вызовите другой метод

// для обработки данных.

private object[,] ExcelScanIntenalOneShet(Workbook workBookIn, int numSheet)

{

// Листы индексируются начиная с 1.

Worksheet sheet = (Worksheet)workBookIn.Sheets[numSheet];

// получение значений

Range excelRange = sheet.UsedRange;

object[,] valueArray = (object[,])excelRange.get\_Value(

XlRangeValueDataType.xlRangeValueDefault);

//

// Сделать что-нибудь с данными в массиве с пользовательским методом.

//

return valueArray;

}

}

}

Можно как-то прокомментировать, конечно, но в принципе оно не особо нужно, просто потом поменяю комментарии сами, сделаю их более понятными

23.06.2017

На данный момент есть вот такой рабочий код, он может «прочитать данные, записать массив начиная с указанной ячейки, и открыть файл». Это ПРОГРЕСС ЧЁРТ ПОБЕРИ!!! Круто!! Теперь можно к самим алгоритмам переходить

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Microsoft.Office.Interop.Excel;

using System.Runtime;

namespace Spektr\_Analis\_sravn\_medods

{

// Этот класс содержит код Excel Interop нам нужно.

// Его основное целью является упрощение работы с Excel, чтобы не заморачиваться во внешнем коде,

// а только вызывать нужные операции, которые всё сами сделают. Я прекрасно понимаю, что это может замедлить программу.

class ExcelReaderInterop

{

// Сохранение объект Application, чтобы можно было использовать в дальнейших функциях/методах.

Application \_excelApp;

// Инициализируется новый читатель Excel. И интегрируется с объектом интерфейса Excel.

public ExcelReaderInterop()

{

\_excelApp = new Application();

}

// Метод, который открывает книги Excel и считывает данные из них в двумерный массив из "объектов"

// Работает без нареканий

public object[,] ExcelReadTableOneShets(string thisFileName, int numSheet)

{

try

{

// открытие

//

// Этот беспорядок кода открывает книгу Excel. Я не знаю, что все

// эти аргументы делают, но они могут быть изменены, чтобы влиять на поведение.

Workbook workBook = \_excelApp.Workbooks.Open(thisFileName,

Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing);

// обращение к внутреннему методу чтения

object[,] valueTable = ExcelScanIntenalOneShet(workBook, numSheet);

// закрытие книги

\_excelApp.Quit();

// возврат полученных значений, используется доп. переменная, чтобы можно было закрыть книгу

return valueTable;

}

catch

{

//

// Обработка исключений.

// В случае ошибки возвращает "null"

//

return null;

}

}

// Метод, который открывает пользователю книгу Excel по указанному адресу, чтобы он мог заносить туда данные

// Работает без нареканий

public void ExcelOpenVisable(string thisFileName)

{

try

{

// открытие

//

// Этот беспорядок кода открывает книгу Excel. Я не знаю, что все

// эти аргументы делают, но они могут быть изменены, чтобы влиять на поведение.

Workbook workBook = \_excelApp.Workbooks.Open(thisFileName,

Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing);

// Собственно делаем видимым

\_excelApp.Visible = true;

}

catch

{

//

// Обработка исключений.

//

}

}

// Метод, который читает лист1 (ОДИН ЛИСТ), внутренний метод

// Сканирование выбранной книги Excel и сохранение информацию об ячейках этой книги в качестве объекта [,] массив.

// Работает без нареканий

private object[,] ExcelScanIntenalOneShet(Workbook workBookIn, int numSheet)

{

// Листы индексируются начиная с 1.

Worksheet sheet = (Worksheet)workBookIn.Sheets[numSheet];

// получение значений

Range excelRange = sheet.UsedRange;

// Не могу прокомментировать именно это решение, так как не особо в нём разбираюсь, могу лишь сказать,

// что оно работает и работает достаточно быстро по сравнению с другими, поэтому и был оставлен именно он.

object[,] valueArray = (object[,])excelRange.get\_Value(

XlRangeValueDataType.xlRangeValueDefault);

return valueArray;

}

// Метод, который записывает книги Excel: C #

// Работает!

private Sheets sheets;

private Worksheet worksheet;

private Range cells;

public void ExcelWriterTableOneShets(string thisFileName, int numSheet, string cels0, Double [,] mas) // для mas (Строка,Столбец)

{

try

{

// Этот беспорядок кода открывает книгу Excel. Я не знаю, что все

// эти аргументы делать, но они могут быть изменены, чтобы влиять на поведение.

\_excelApp = new Application();

Workbook workbook = \_excelApp.Workbooks.Open(thisFileName,

Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing);

sheets = workbook.Worksheets;

//Получаем ссылку на лист "numSheet"

worksheet = (Worksheet)sheets.get\_Item(numSheet);

// Еще один способ определения выбранных ячеек - использование метода get\_Offset(x,y) объекта Range,

// возвращающего объект Range, отстоящий от заданной ячейки на заданное количество строк и столбцов,

// считая от левого верхнего угла. Это позволяет работать с ячейками, позиция которых заданы относительно

// выбранной ячейки или группы ячеек.

cells = worksheet.get\_Range(cels0, cels0);

int a = mas.GetLength(0);

int b = mas.GetLength(1);

for (int i = 0; i < a; i++) // цикл выполнится 5 раз

{

cells = cells.get\_Offset(i, 0);

for (int j = 0; j < b; j++) // цикл выполнится 5 раз

{

cells.Value2 = mas[i, j];

cells = cells.get\_Offset(0, 1);

}

cells = worksheet.get\_Range(cels0, cels0);

}

workbook.Save();

\_excelApp.Quit();

}

catch

{

//

// Обработка исключений.

//

}

}

}

}

24.06.2017

Что важно нужно заметить? То что я смог реализовать запись в файл, вот правда не нравится мне в этом следующее, долго он работает, долго, да, наверное у меня не лучший способ реализации, да и пофиг. Сейчас сначала размещу код, который реализует эту запись (стоит заметить, я сделал этот метод перегруженным, то есть там 2 таких экземпляра, чтобы можно было не только массивы, но и одиночные цифры записывать, если возникнет необходимость, нужно будет то же самое реализовать и для строковых типов)

// Метод, который записывает книги Excel #

// Работает!

private Sheets sheets;

private Worksheet worksheet;

private Range cells;

public void ExcelWriterTableOneShets(string thisFileName, int numSheet, string cels0, Double [,] mas) // для mas (Строка,Столбец)

{

try

{

// Этот беспорядок кода открывает книгу Excel. Я не знаю, что все

// эти аргументы делать, но они могут быть изменены, чтобы влиять на поведение.

\_excelApp = new Application();

Workbook workbook = \_excelApp.Workbooks.Open(thisFileName,

Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing);

sheets = workbook.Worksheets;

//Получаем ссылку на лист "numSheet"

worksheet = (Worksheet)sheets.get\_Item(numSheet);

// Еще один способ определения выбранных ячеек - использование метода get\_Offset(x,y) объекта Range,

// возвращающего объект Range, отстоящий от заданной ячейки на заданное количество строк и столбцов,

// считая от левого верхнего угла. Это позволяет работать с ячейками, позиция которых заданы относительно

// выбранной ячейки или группы ячеек.

cells = worksheet.get\_Range(cels0, cels0);

int a = mas.GetLength(0); // количество строк

int b = mas.GetLength(1); // количество столбцов

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

{

cells = cells.get\_Offset(i, 0); //смещение относительно базовой на iстрок

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

{

cells.Value2 = mas[i, j];

cells = cells.get\_Offset(0, 1); //смещение относительно заданой выше на j столбцов

}

cells = worksheet.get\_Range(cels0, cels0);

}

// сохранение книги

workbook.Save();

\_excelApp.Quit();

}

catch

{

//

// Обработка исключений.

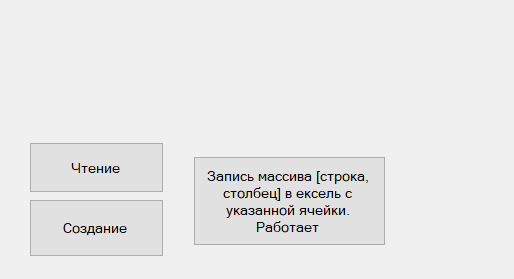
//

}

}

Комментировать я особо не будут, так как там по ходу всё что нужно прокомментировано.

Так же вчера я реализовал создание синусоидального сигнала.



Вот по сути этот кусок, никакого дельного оформления ещё нет, так что на это можно и не обращать внимания, главное, что оно работает. Далее выкладываю код кнопок, комментарии там тоже излишне, ибо всё и так расписано.

Кнопка «чтения»

private void button1\_Click(object sender, EventArgs e)

{

ExcelReaderInterop reader = new ExcelReaderInterop();

object[,] parametrs = reader.ExcelReadTableOneShets(Application.StartupPath + @"\ParamerSign.xlsx", 1);

f\_osn = Convert.ToDouble(parametrs[1, 1]);

f\_fact = Convert.ToDouble(parametrs[2, 1]); // счтывание параметров

N\_onePeriod = Convert.ToInt32(parametrs[3, 1]);

n\_Periodov = Convert.ToInt32(parametrs[4, 1]);

int nStrok = parametrs.GetLength(0); // количество строк

sign\_param = new double[nStrok-5, 3]; // стоит иметь ввиду, что если строк 5, то последня 4-ая, так как нумерация с 0

for (int i = 6; i < nStrok; i++)

{

sign\_param[i - 6, 0] = Convert.ToDouble(parametrs[i, 1]);

sign\_param[i - 6, 1] = Convert.ToDouble(parametrs[i, 2]);

sign\_param[i - 6, 2] = Convert.ToDouble(parametrs[i, 3]);

}

}

Кнопка «создание»

private void button4\_Click(object sender, EventArgs e)

{

int nGarmon = sign\_param.GetLength(0); // определяется количество гармоник

signal = new double[n\_Periodov\*N\_onePeriod, 2]; // сразу иницилизируется матрица сигнала

if (sign\_param[0, 0] == 0)

{

for (int i = 0; i < n\_Periodov \* N\_onePeriod; i++)

{

signal[i, 0] = i / f\_osn / N\_onePeriod;

signal[i, 1] = sign\_param[0, 1];

for (int j = 1; j < nGarmon; j++)

{

signal[i, 1] = signal[i, 1] + sign\_param[j, 1]\*Math.Sin(2\*Math.PI\*f\_fact\* sign\_param[j, 0]\* signal[i, 0] + sign\_param[j, 0]\*Math.PI/180);

//в этой формуле амплитуда \* sin ( 2\*pi \*f\_fact \* ngarmon \* t + fi0)

}

}

}

else

{

for (int i = 0; i < n\_Periodov \* N\_onePeriod; i++)

{

signal[i, 0] = i / f\_osn / N\_onePeriod;

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

{

signal[i, 1] = signal[i, 1] + sign\_param[j, 1] \* Math.Sin(2 \* Math.PI \* f\_fact \* sign\_param[j, 0] \* signal[i, 0] + sign\_param[j, 0] \* Math.PI / 180);

//в этой формуле амплитуда \* sin ( 2\*pi \*f\_fact \* ngarmon \* t + fi0)

}

}

}

}

И собственно кнопка «запись в массив» - тут замечу вот что, она заполняет по шаблону, поэтому и вручную прописываются начальные ячейки

private void button2\_Click(object sender, EventArgs e)

{

ExcelReaderInterop writer = new ExcelReaderInterop();

writer.ExcelWriterTableOneShets(Application.StartupPath + @"\Signal.xlsx", 1, "A1", f\_osn);

writer.ExcelWriterTableOneShets(Application.StartupPath + @"\Signal.xlsx", 1, "A2", f\_fact);

writer.ExcelWriterTableOneShets(Application.StartupPath + @"\Signal.xlsx", 1, "A3", N\_onePeriod);

writer.ExcelWriterTableOneShets(Application.StartupPath + @"\Signal.xlsx", 1, "A4", n\_Periodov);

writer.ExcelWriterTableOneShets(Application.StartupPath + @"\Signal.xlsx", 1, "A6", signal);

}

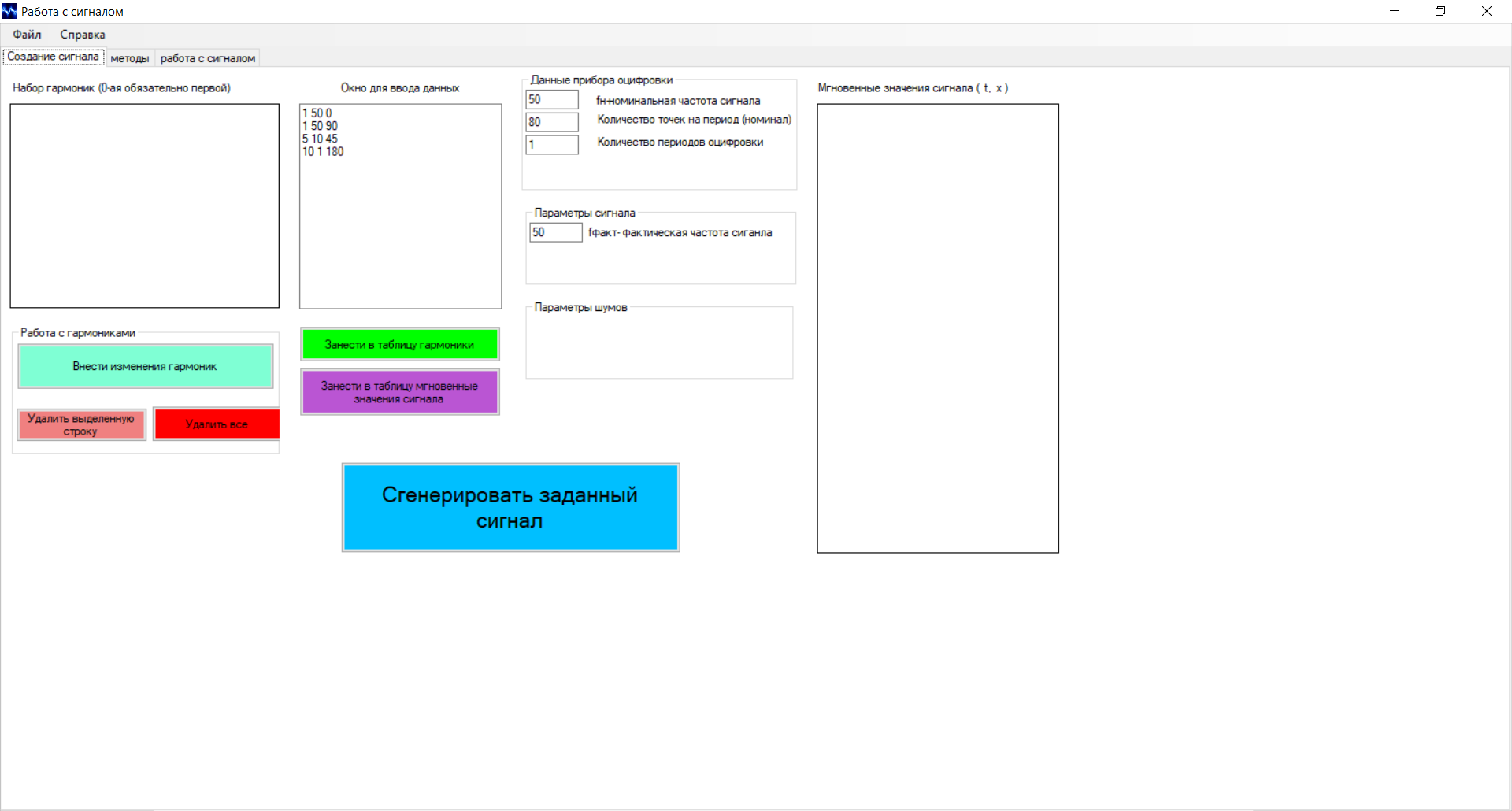
Так же замечу вот что. Сейчас я буду переделывать оформление, делать его адекватным, поэтому многое может измениться, причём довольно сильно

# Реализация формы «создание сигнала»

24.06.2017

Вообще я сегодня по сути именно этим и занимался. По сути всё что делал с екселем – нафиг не нужно, НО! Оно нужно, ибо благодаря этому я научился работать хоть как-то с ним и вдоволь поматерился

Сегодня я занимался другим, а если конкретнее – то работал с базами, туго у меня это пошло, но я чего-то да добился. Я не стал использовать базы, но вот формы я реализовал вполне нормально. А вот с базами поматерился. Иии…. По началу оно работало, но у меня не получалось нормально вставлять в эту форму… и пока я пытался это сделать я столько дров наломал… В итоге сделал иначе добавил текстовое окошко, через которое можно это всё сделать… Ну это конечно криво, но зато работает. Всё сейчас выглядит вот так



Ну не супер, ну и ладно, это уже ОЧЕНЬ круто и оно работает. Теперь добавить нужные блоки бы с методами и т.д. Вот этим завтра и займусь. Не особо нужно там много чего реализовывать, по оформлению и сравнению имею ввиду, ибо это всё нужно будет обсудить ещё. А вот сами методы запихнуть, это да, это нужно + разобраться в них. А сейчас скину сюда код всего этого класса, знаю, знаю, что кривовато так делать, но мне «маленькому программисту можно»

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Spektr\_Analis\_sravn\_medods

{

public partial class WorkSignal : Form

{

// задание существенных параметров сигнала, реализованы публичными, так как могут использоваться и вне класса - на данный момент точно не известно

public double[,] paramets\_table; // параметры гармоник

public double[,] signal\_table; // значения массива сигнала

public double fn; // номинальная частота сигнала

public double f\_fact; // фактическая частота сигнала

public int N\_one\_period; // количество точек на период

public int n\_periods; // количество периодов оцифровки

public WorkSignal()

{

InitializeComponent();

// задание нулевых значений существенных параметров

fn = Convert.ToDouble(fn\_TB.Text);

f\_fact = Convert.ToDouble(f\_fact\_TB.Text);

N\_one\_period = Convert.ToInt32(N\_one\_period\_TB.Text);

n\_periods = Convert.ToInt32(n\_Period\_TB.Text);

}

private void WorkSignal\_Load(object sender, EventArgs e)

{

}

private void DeletOne\_param\_button\_Click(object sender, EventArgs e)

{

int ind = parametrs\_DGV.SelectedCells[0].RowIndex;

parametrs\_DGV.Rows.RemoveAt(ind);

}

private void DeletAll\_param\_button\_Click(object sender, EventArgs e)

{

while (parametrs\_DGV.Rows.Count > 1)

for (int i = 0; i < parametrs\_DGV.Rows.Count - 1; i++)

parametrs\_DGV.Rows.Remove(parametrs\_DGV.Rows[i]);

}

// запись в таблицу гармоник, код будет рассмотрен для этого случая, для сигнала всё аналогично

private void Write\_button\_Click(object sender, EventArgs e)

{

paramets\_table = null; // обнуление, если уже были параметры, для надёжности

byte col\_stolb = 3; //задаём колличество столбцов

int n\_strok = (int)Vstavka\_TB.Lines.Length; //определяем колличество строк в входном массиве

string[] stroki = Vstavka\_TB.Lines; // передаём построчно массив

string[] chisla\_prov = stroki[0].Split(); // разбиваем строки на символы (разделитель - пробел)

if (chisla\_prov.Length == col\_stolb) // // обработка ошибки (если неверно скопированно)

{

chisla\_prov = stroki[n\_strok - 1].Split(); // разбиваем строки на символы (разделитель - пробел)

if (chisla\_prov.Length != col\_stolb)

{

n\_strok = n\_strok - 1; // обработка ошибки (неверная строка, когда элементов в строке не столько, сколько должно быть)

}

paramets\_table = new double[n\_strok, col\_stolb]; // создаём массив для обработки строк

for (int i = 0; i < n\_strok; i++) // перебор по строкам

{

string[] chisla = stroki[i].Split(); // разбиваем строки на символы (разделитель - пробел)

for (int j = 0; j < chisla.Length; j++) // перебор по столбцам

{

paramets\_table[i, j] = Convert.ToDouble(chisla[j]); //запись числа

}

}

// предварительная очистка

while (parametrs\_DGV.Rows.Count > 1)

for (int i = 0; i < parametrs\_DGV.Rows.Count - 1; i++)

parametrs\_DGV.Rows.Remove(parametrs\_DGV.Rows[i]);

// создания пустых строк

parametrs\_DGV.ColumnCount = col\_stolb;

// название столбцов

parametrs\_DGV.Columns[0].HeaderText = "Номер гармоники";

parametrs\_DGV.Columns[1].HeaderText = "Амплитула";

parametrs\_DGV.Columns[2].HeaderText = "Фаза (град)";

parametrs\_DGV.Rows.Add(n\_strok);

// запись

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

{

parametrs\_DGV.Rows[i].Cells[0].Value = paramets\_table[i, 0];

parametrs\_DGV.Rows[i].Cells[1].Value = paramets\_table[i, 1];

parametrs\_DGV.Rows[i].Cells[2].Value = paramets\_table[i, 2];

}

}

}

// заполнение таблицы с сигналом - Если уже есть данные

private void write\_sinal\_button\_Click(object sender, EventArgs e)

{

signal\_table = null; // обнуление, если уже были параметры, для надёжности

byte col\_stolb = 2;

int n\_strok = (int)Vstavka\_TB.Lines.Length;

string[] stroki = Vstavka\_TB.Lines;

string[] chisla\_prov = stroki[0].Split(); // разбиваем строки на символы (разделитель - пробел)

if (chisla\_prov.Length == col\_stolb) // обработка ошибки (если неверно скопированно)

{

chisla\_prov = stroki[n\_strok - 1].Split(); // разбиваем строки на символы (разделитель - пробел)

if (chisla\_prov.Length != col\_stolb)

{

n\_strok = n\_strok - 1; // обработка ошибки (неверная строка, когда элементов в строке не столько, сколько должно быть)

}

signal\_table = new double[n\_strok, col\_stolb];

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

{

string[] chisla = stroki[i].Split();

for (int j = 0; j < chisla.Length; j++)

{

signal\_table[i, j] = Convert.ToDouble(chisla[j]);

}

}

// предварительная очистка

while (Signal\_dGV.Rows.Count > 1)

for (int i = 0; i < Signal\_dGV.Rows.Count - 1; i++)

Signal\_dGV.Rows.Remove(Signal\_dGV.Rows[i]);

// создания пустых строк

Signal\_dGV.ColumnCount = col\_stolb;

Signal\_dGV.Columns[0].HeaderText = "время (с)";

Signal\_dGV.Columns[1].HeaderText = "Мгновенное значение";

Signal\_dGV.Rows.Add(n\_strok);

// запись

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

{

Signal\_dGV.Rows[i].Cells[0].Value = signal\_table[i, 0];

Signal\_dGV.Rows[i].Cells[1].Value = signal\_table[i, 1];

}

}

}

private void fn\_TB\_TextChanged(object sender, EventArgs e)

{

fn = Convert.ToDouble(fn\_TB.Text);

}

private void N\_one\_period\_TB\_TextChanged(object sender, EventArgs e)

{

N\_one\_period = Convert.ToInt32(N\_one\_period\_TB.Text);

}

private void n\_Period\_TB\_TextChanged(object sender, EventArgs e)

{

n\_periods = Convert.ToInt32(n\_Period\_TB.Text);

}

private void f\_fact\_TB\_TextChanged(object sender, EventArgs e)

{

f\_fact = Convert.ToDouble(f\_fact\_TB.Text);

}

// создание сигнала

private void MadeSignal\_button\_Click(object sender, EventArgs e)

{

if (paramets\_table != null)

{

signal\_table = null; // обнуление, если уже были параметры, для надёжности

int nGarmon = paramets\_table.GetLength(0); // определяется количество гармоник

signal\_table = new double[N\_one\_period \* n\_periods, 2]; // сразу иницилизируется матрица сигнала

if (paramets\_table[0, 0] == 0)

{

for (int i = 0; i < N\_one\_period \* n\_periods; i++)

{

signal\_table[i, 0] = 0; //принудительное обнуление, на всякий случай

signal\_table[i, 1] = 0; //принудительное обнуление, на всякий случай

signal\_table[i, 0] = i / fn / N\_one\_period;

signal\_table[i, 1] = paramets\_table[0, 1];

for (int j = 1; j < nGarmon; j++)

{

signal\_table[i, 1] = signal\_table[i, 1] + paramets\_table[j, 1] \* Math.Sin(2 \* Math.PI \* f\_fact \* paramets\_table[j, 0] \* signal\_table[i, 0] + paramets\_table[j, 2] \* Math.PI / 180);

//в этой формуле амплитуда \* sin ( 2\*pi \*f\_fact \* ngarmon \* t + fi0)

}

}

}

else

{

for (int i = 0; i < N\_one\_period \* n\_periods; i++)

{

signal\_table[i, 0] = 0; //принудительное обнуление, на всякий случай

signal\_table[i, 1] = 0; //принудительное обнуление, на всякий случай

signal\_table[i, 0] = i / fn / N\_one\_period;

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

{

signal\_table[i, 1] = signal\_table[i, 1] + paramets\_table[j, 1] \* Math.Sin(2 \* Math.PI \* f\_fact \* paramets\_table[j, 0] \* signal\_table[i, 0] + paramets\_table[j, 2] \* Math.PI / 180);

//в этой формуле амплитуда \* sin ( 2\*pi \*f\_fact \* ngarmon \* t + fi0)

}

}

}

// запись в столбец

byte col\_stolb = 2;

// предварительная очистка

while (Signal\_dGV.Rows.Count > 1)

for (int i = 0; i < Signal\_dGV.Rows.Count - 1; i++)

Signal\_dGV.Rows.Remove(Signal\_dGV.Rows[i]);

// создания пустых строк

Signal\_dGV.ColumnCount = col\_stolb;

Signal\_dGV.Columns[0].HeaderText = "время (с)";

Signal\_dGV.Columns[1].HeaderText = "Мгновенное значение";

Signal\_dGV.Rows.Add(N\_one\_period \* n\_periods);

// запись

for (int i = 0; i < N\_one\_period \* n\_periods; i++)

{

Signal\_dGV.Rows[i].Cells[0].Value = signal\_table[i, 0];

Signal\_dGV.Rows[i].Cells[1].Value = signal\_table[i, 1];

}

}

}

private void ObnovlParametrs\_button\_Click(object sender, EventArgs e)

{

// определяем количество строк

int n\_strok = parametrs\_DGV.Rows.Count - 1;

paramets\_table = null; // обнуление, если уже были параметры, для надёжности

byte col\_stolb = 3; //задаём колличество столбцов

// иницилизируем массив

paramets\_table = new double[n\_strok, col\_stolb]; // создаём массив для обработки строк

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

{

paramets\_table[i, 0] = Convert.ToDouble(parametrs\_DGV.Rows[i].Cells[0].Value);

paramets\_table[i, 1] = Convert.ToDouble(parametrs\_DGV.Rows[i].Cells[1].Value);

paramets\_table[i, 2] = Convert.ToDouble(parametrs\_DGV.Rows[i].Cells[2].Value);

}

}

}

}

Вот странно… не хочет окраску сохранять… ну и ладно.

А вообще прикольно это, по сути дело пошло только ближе к вечеру ночи, ибо по началу я всё таки пытался настроит эти базы, но потом забил и сделал иначе))) Криво, но рабоче))

25.06.2017 вечер

Думаю расписывать каждое действие можно бы и прекратить, буду расписывать более глобально.

Начал я создавать алгоритмы сначала долго копался с методами связанными с комплексными числами… в общем не заработало оно толком, и хрен его знает, почему так… как-то уж совсем не работало.

Вообще – нужно будет подумать реализации своего класса адекватного, ибо встроенный – это пиздец (а точнее можно бы и найти, я там нормальные видел, нужно будет только под себя переделать)

Касательно самого метода. Хорошо расписано и сделано у Андрея, буду дальше по ним разбираться. У себя я только ДПФ обычный реализовал. Он у меня почему-то неверно работал, я долго пытался понять, пока не прогнал его нормально, и тогда уже понял

Теперь немного о эмоциях, это сообщения, которые я писал

«Какие порой фокусы случаются по типу "вначале не то значение взял"....

.

Я опять про программирование, а суть вот в чём

предположим у тебя есть А и В

В - расчитывается отдельно для каждой точки

А = f(В), НО!! В-то уже расчитанно, то есть просто цифирки подставить.

А я сделал так

А = f (В - которая каждый раз снова считается)

.

А я то-то думаю, что у меня долго так считается когда точек побольше, а это .... Кхмм... ну в общем... Мать его - геометрическая прогрессия и 2 в 2 это мало, а вот 2 в 10 это уже 1024, а у меня стпепени поболее были.... и не для двойки...

.

.

В общем не только в курсачах КПД более 100 получается)) (у меня конечно не КПД считается. но мне нужно оценивать скорость и точность, а я такой фокус со скоростью устроил)»

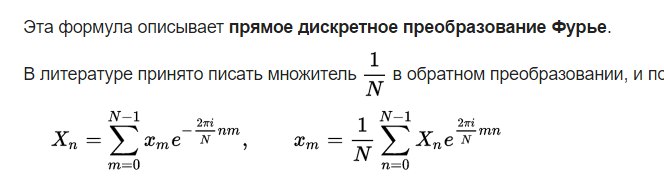
«в общем сейчас ряд фурье

Вот вторая формула, там внутри считается сумма, и это при том, что сколько элементов в сумме, столько и гармоник.

То есть, если у тебя 10 точек, то ты можешь выделить 10 гармоник, и того у тебя уже 10 переборов по 10 раз.

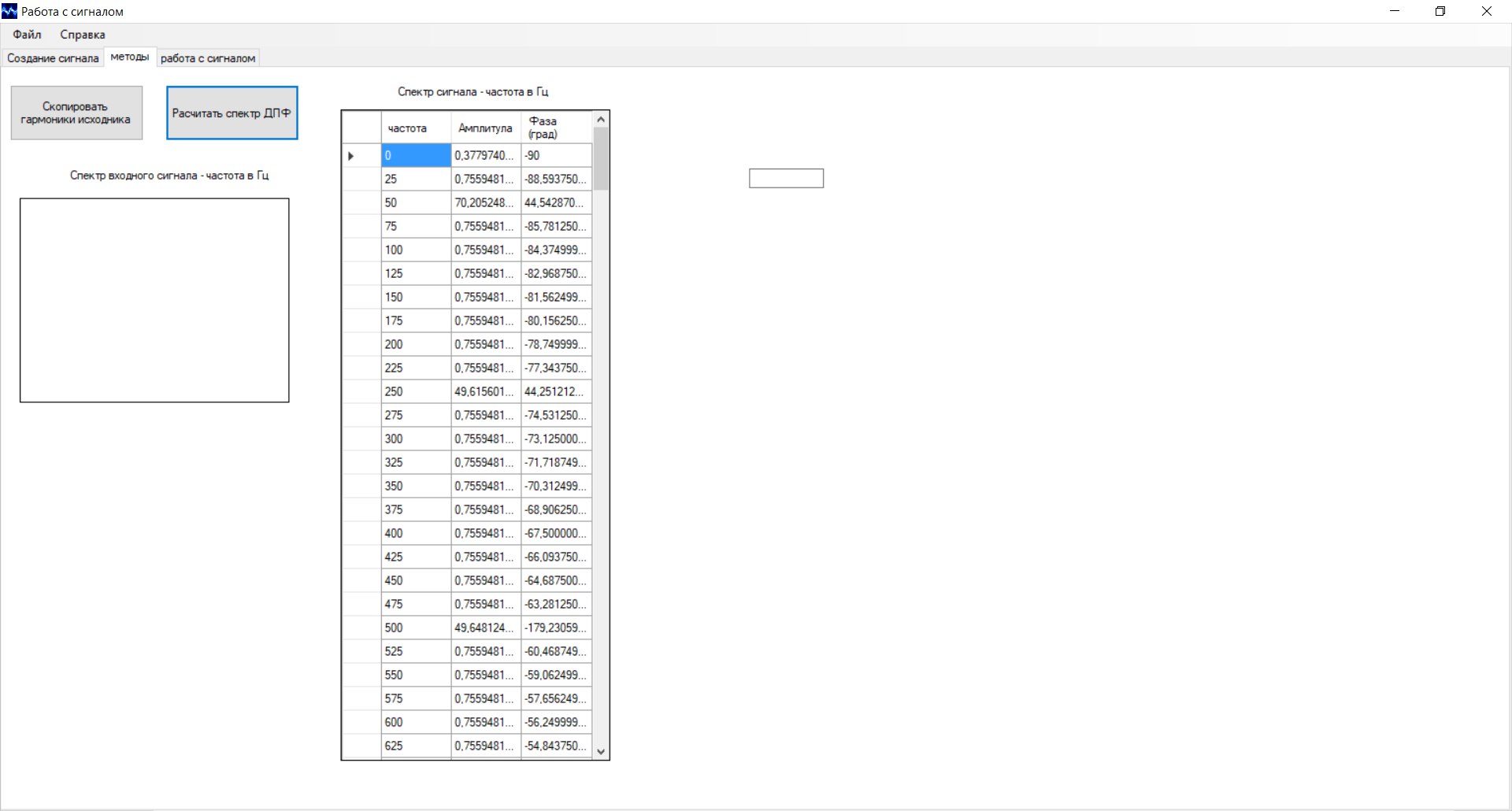
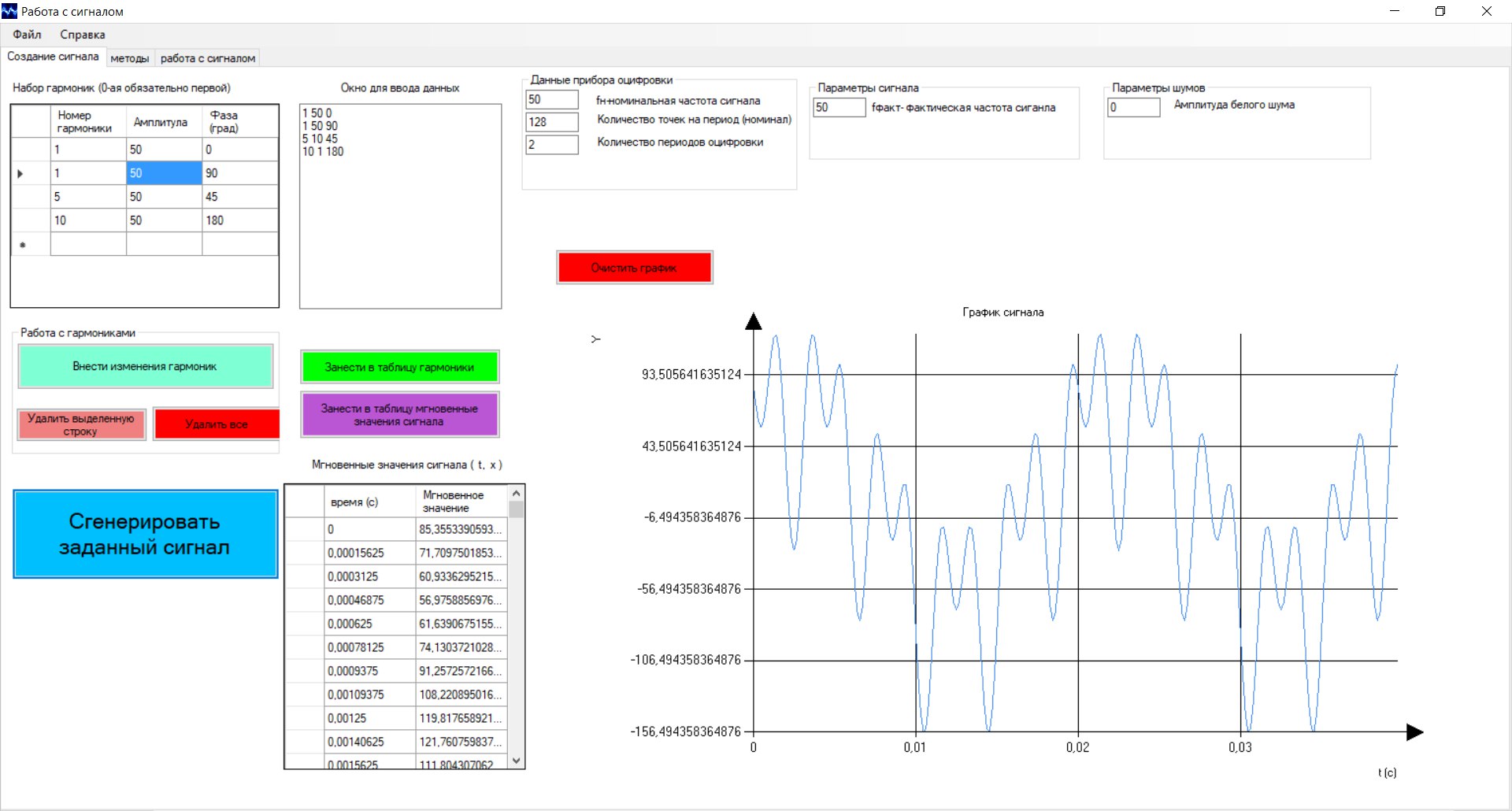
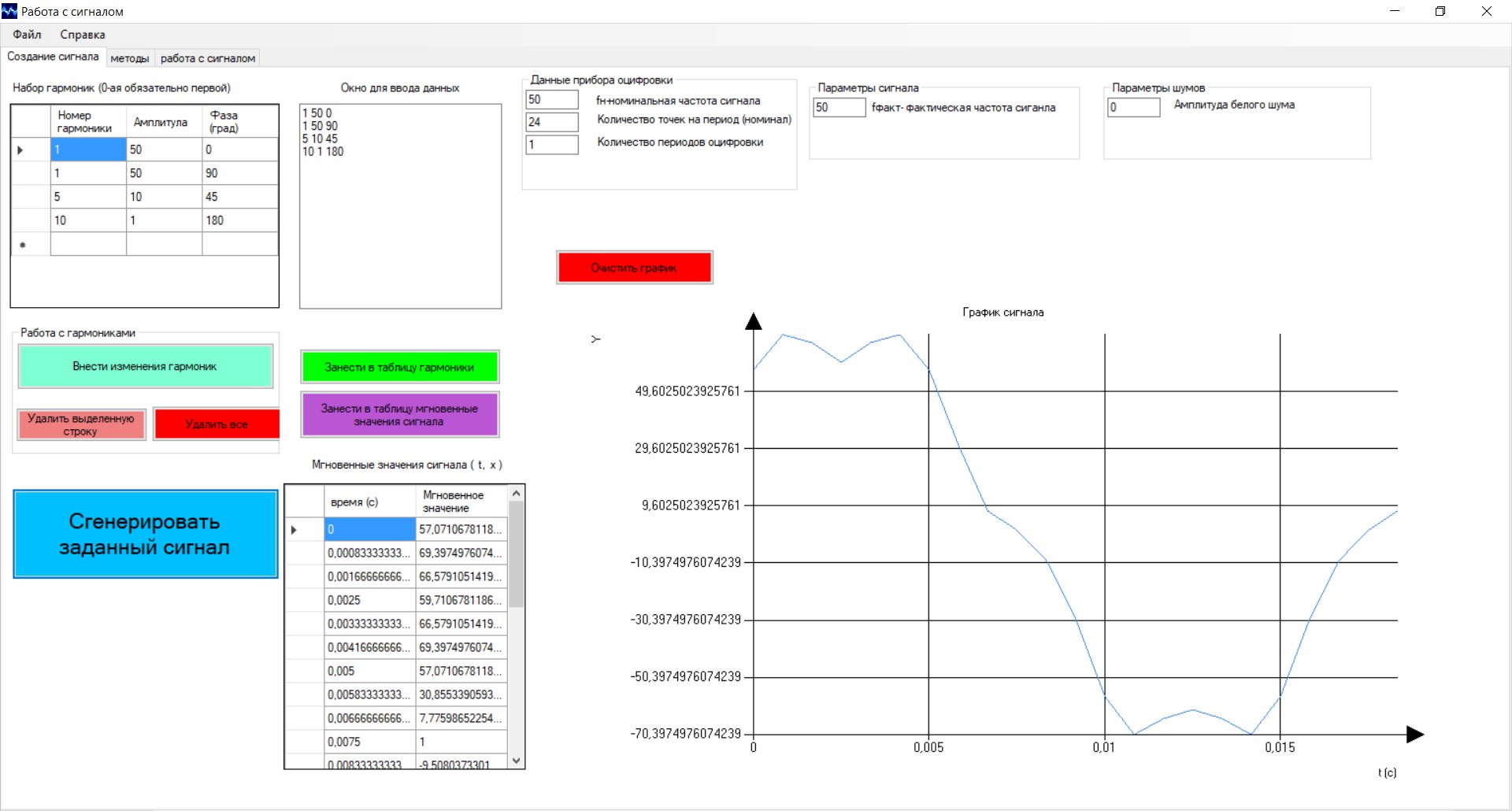
А я внутрь запихнул ещё один перебор.

то есть тут было n^2, а я сделал n^(3-4)»



И ещё немного скринов из программы

Касательно шумов, на данный момент я реализовал только один, и его в принципе достаточно



Завтра вот буду подойду на кафедру, поговорю о дальнейшей модернизации всего этого