Содержание

[ВВЕДЕНИЕ 5](#_Toc6564254)

[1 Математическая модель 6](#_Toc6564255)

[1.1 Критическое затухание 6](#_Toc6564256)

[1.2 Недодемпфированные системы (с затуханием ниже критического). 7](#_Toc6564257)

[1.3 Передемпфированные системы (с затуханием больше критического). 8](#_Toc6564258)

[2 Описание программы 9](#_Toc6564259)

[2.1 Структурная схема программы 9](#_Toc6564260)

[2.2 Описание разработанного класса 9](#_Toc6564261)

[2.3 Описание модуля графического и табличного представления данных 11](#_Toc6564262)

[2.4 Описание метода импорта в Excel 13](#_Toc6564263)

[2.5 Возможности проекта 15](#_Toc6564264)

[3 Руководство пользователя 16](#_Toc6564265)

[4 Методика испытания 20](#_Toc6564266)

[ЗАКЛЮЧЕНИЕ 23](#_Toc6564267)

[Список используемой литературы 24](#_Toc6564268)

[Приложение А 25](#_Toc6564269)

# ВВЕДЕНИЕ

Программное моделирование анализа свободных затухающих колебаний линейного осциллятора является весьма распространенным явлением при эксплуатации техники, строительных сооружений и в других областях народного хозяйства. Осцилляторы в основном преобразовывают однонаправленный ток из источника постоянного тока в переменную форму волны, которая имеет желательную частоту.

Данная программа позволяет производить расчетсвободных затухающих колебаний линейного осциллятора, отображать полученную информацию в наглядном виде, облегчающем понимание и осмысление происходящих в системе колебательных процессов.

Для разработки программы была выбрана среда визуального проектирования Microsoft Visual Studio 2017. Она позволяет за короткий промежуток времени и с наименьшими затратами физических и умственных сил создавать прикладные программы различной направленности, обеспечивая при этом наличие дружественного оконного интерфейса. Указанные (а также ряд других) преимущества среды Microsoft Visual Studio 2017, обуславливают широкое её использование при создании программного обеспечения разнообразного назначения.

Цель данной работы продемонстрировать навыки, полученные при работе с программной средой Visual Studio на примере программы, реализующей моделирование анализа свободных затухающих колебаний линейного осциллятора.

# 1 Математическая модель

Колебания системы называются свободными, если скорость изменения состояния системы определяется только состоянием самой системы, а именно восстанавливающей равновесное состояние силой, зависящей от величины обобщенной координаты, которая определяет отклонение системы из этого состояния. Такая система называется линейным осциллятором.

Если в системе присутствует затухание, то решение уравнения, определяющего реакцию системы, имеет вид

|  |  |  |
| --- | --- | --- |
|  |  | (1.1) |

Это формула характеризует три вида движения в зависимости от знака выражения под квадратным корнет, т.е. если эта величина положительная, отрицательная или равна нулю. Сначала удобно рассмотреть предельный случай, когда выражение под квадратным корнем равна нулю. Этот случай определяется условие критического затухания.

## **Критическое затухание**

Если выражение под квадратным корнем становится равным нулю, то очевидно, что; таким образом, величина критического затухания равна:

|  |  |  |
| --- | --- | --- |
|  |  | (1.2) |

В этом случае величина s в выражении

|  |  |  |
| --- | --- | --- |
|  |  | (1.3) |

а реакция в соответствии с уравнением () равна:

|  |  |  |
| --- | --- | --- |
|  |  | (1.4) |

где второй член умножается на t, так как в решении уравнения присутствует только одно значение s ().

Постановка начальных условий в уравнение () приводит к окончательной форме уравнения реакции для случая критического затухания:

|  |  |  |
| --- | --- | --- |
|  |  | (1.5) |

Видно, что реакция при свободных колебаниях системы с критическим затуханием не содержит колебаний относительно положения нулевого перемещения. Напротив, перемещение стремится к нулю в соответствии с членом уравнения, характеризующим затухание колебаний по экспоненте.

Ценность формулировки состояния системы с критическим затуханием заключается в том, что оно определяет минимальную величину затухания, при котором не происходит колебаний в стадии свободных колебаний реакции системы.

## Недодемпфированные системы (с затуханием ниже критического).

Если затухание меньше критического, то из уравнения (1.2) видно, что , и, таким образом, подкоренное выражение в формуле (1.1) должно быть отрицательным. Для определения реакции при свободных колебаниях в этом случае удобно выразить затухание как отношение *с* к величине критического затухания *сс*, таким образом

|  |  |  |
| --- | --- | --- |
|  |  | (1.6) |

в котором называется параметром затухания. Подставляя выражение (1.6) в (1.1), получаем или, изменяя знак подкоренного выражения и вводя новое обозначение , имеем:

|  |  |  |
| --- | --- | --- |
|  |  | (1.7) |

где

|  |  |  |
| --- | --- | --- |
|  |  | (1.8) |

Величина называется частотой демпфированных колебаний, или частотой колебаний с учетом затухания. Для параметров затухания, характерных для обычных строительных конструкций ( <20%), частота , как следует из (1.8), очень незначительно отличается от частоты недемпфированных колебаний. Чтобы определить влияние затухания на частоту, следует помнить, что график зависимости отношения частоты колебаний с учетом затухания к частоте незатухающих колебаний от параметра затухания представляет собой окружность, радиус которой равен единице.

Путем подстановки выражения (1.7) можно определить реакцию недодемпфированной системы при свободных колебаниях

|  |  |  |
| --- | --- | --- |
|  |  | (1.9) |

В этом уравнении член в круглых скобках характеризует простое гармоническое движение. Таким образом, это уравнение можно записать в более удобной форме

|  |  |  |
| --- | --- | --- |
|  |  | (1.10) |

И, наконец, при введении начальных условий и можно определить постоянные в (1.10).

|  |  |  |
| --- | --- | --- |
|  |  | (1.11) |

## Передемпфированные системы (с затуханием больше критического).

Хотя в обычных условиях строительные системы, имеющие затухание больше критического, не встречаются, для полноты исследования полезно рассмотреть анализ реакции передемпфированной системы. В этом случае ξ>1, и выражение (1.1) можно представить в виде:

|  |  |  |
| --- | --- | --- |
|  |  | (1.12) |

Подстановка выражения (1.12) позволяет после преобразований получить:

|  |  |  |
| --- | --- | --- |
|  |  | (1.13) |

Здесь константы А и В можно вычислить с учетом начальных условий. Видно, что движение передемпфированной системы не является колебательным. Оно аналогично реакции системы с критическим, но возврат в нейтральное положение замедляется с ростом параметра затухания

# 2 Описание программы

## 2.1 Структурная схема программы

Структурная схема программы состоит из главного модуля, модулей описания вспомогательных форм, динамических библиотек, файлов помощи и обращений к другим показана на рисунке 1.



Рисунок 1 – Структурная схема

Структурная схема показывает взаимодействие объектов класса. После загрузки формы все основные действия происходят в главном окне. Главное окно запрашивает определенные действия с помощью определенных элементов управления на форме.

## 2.2 Описание разработанного класса

В данном приложение был разработан главный класс со всеми необходимыми функция для подсчета и обработки информации по данной теме курсового проекта. Данный класс назван «Phisics.cs»

//Вводимые параметры

public double m { get; set; } // масса

public int m\_item { get; set; }

public double q0 { get; set; } //начальное положение

public int q0\_item { get; set; }

public double dq0 { get; set; } //начальная скорость

public int dq0\_item { get; set; }

public double k { get; set; } // коэфициент жесткости пружины????

public int k\_item { get; set; }

public double c { get; set; } // коэфициент затухания

public int c\_item { get; set; }

public double time { get; set; } // время измерения

public int time\_item { get; set; }

public double time\_interval { get; set; } // интервал измерения

public int time\_interval\_item { get; set; }

//Неизвестные

private double n //

{

get

{

return c / (2 \* m);

}

}

public double w // круговая частота свободного колебания

{

get

{

return Math.Sqrt(k / m);

}

}

public double E // параметр затухания

{

get

{

return c / 2 \* m \* w;

}

}

private double Wd // частота колебания с затуханием или чаcтота демпфированных колебаний

{

get

{

return w \* Math.Sqrt(1 - Math.Pow(E, 2));

}

}

private double WV

{

get

{

return w \* Math.Sqrt(Math.Pow(E, 2) - 1);

}

}

public double q(double t)

{

double result = 0;

if (n == w) // критическое затухание

result = (q0 \* (1 + w \* t) + dq0 \* t) \* Math.Pow(Math.E, -1 \* w \* t);

if (n < w) // недодемпфированные системы ( с затуханием ниже критического)

result = Math.Pow(Math.E, -1 \* w \* t \* E) \* (((dq0 + q0 \* E \* w) / Wd) \* Math.Sin(Wd \* t) + q0 \* Math.Cos(Wd \* t));

if (n > w) // Передемпфированные системы ( с затуханием больше критического)

result = Math.Pow(Math.E, -1 \* w \* t \* E) \* (((dq0 + q0 \* E \* w) / WV) \* sh(WV \* t) + q0 \* ch(WV \* t));

return result;

}

Можно заметить, параметры, которые известны изначально, являются публичными, так как их значения вводятся и в дальнейшем строится график. Параметры, имеющие статус приватных, высчитываются в самом классе и не могут быть вызваны из вне класса.

## 2.3 Описание модуля графического и табличного представления данных

Для отображения графической составляющей в данном проекте используются класс «Points.cs», который служит для формирования массива точек, которые послужат для отображения графика.

List<Points> listPoints = new List<Points>();

public double X { get; set; }

public double Y { get; set; }

public Points() { }

private Points(double x, double y)

{

this.X = x;

this.Y = y;

}

private void AddPoints(Phisics phisics)

{

for (double time = 0; time < phisics.time; time += phisics.time\_interval)

listPoints.Add(new Points(time, phisics.q(time)));

}

public List<Points> PointsList(Phisics phisics)

{

AddPoints(phisics);

return listPoints;

}

Можно обратить внимание на то, что в данном классе есть два конструктора, один является публичным, а другое – приватным. В данный класс обращаются для получения точек, на основе которого строится график функций.

Для табличного представления данных был разработан класс «Tables.cs».

class Tables

{

public List<DataGridViewTextBoxColumn> CreateTableColumn(Settings settings)

{

List<DataGridViewTextBoxColumn> column = new List<DataGridViewTextBoxColumn>();

if (settings.time\_Check)

{

DataGridViewTextBoxColumn col0 = new DataGridViewTextBoxColumn();

col0.HeaderText = "Текущее время";

col0.Name = "ID";

column.Add(col0);

}

if (settings.mass\_Check)

{

DataGridViewTextBoxColumn col0 = new DataGridViewTextBoxColumn();

col0.HeaderText = "Масса";

col0.Name = "mass";

column.Add(col0);

}

if (settings.q\_Check)

{

DataGridViewTextBoxColumn col0 = new DataGridViewTextBoxColumn();

col0.HeaderText = "Результат функции";

col0.Name = "func";

column.Add(col0);

}

if (settings.dq0\_Check)

{

DataGridViewTextBoxColumn col0 = new DataGridViewTextBoxColumn();

col0.HeaderText = "Скорость";

col0.Name = "dq0";

column.Add(col0);

}

if (settings.c\_Check)

{

DataGridViewTextBoxColumn col0 = new DataGridViewTextBoxColumn();

col0.HeaderText = "Коэффициент затухания";

col0.Name = "c";

column.Add(col0);

}

if (settings.w\_Check)

{

DataGridViewTextBoxColumn col0 = new DataGridViewTextBoxColumn();

col0.HeaderText = "Круговая частота";

col0.Name = "w";

column.Add(col0);

}

if (settings.E\_Check)

{

DataGridViewTextBoxColumn col0 = new DataGridViewTextBoxColumn();

col0.HeaderText = "Параметр затухания";

col0.Name = "e";

column.Add(col0);

}

return column;

}

public List<DataGridViewRow> CreateTableRow(Phisics phisics, Settings settings)

{

List<DataGridViewRow> dataTable = new List<DataGridViewRow>();

for (double t = 0; t < phisics.time; t += phisics.time\_interval)

{

DataGridViewRow row = new DataGridViewRow();

if (settings.time\_Check)

{

DataGridViewCell columnValue = new DataGridViewTextBoxCell();

columnValue.Value = t;

row.Cells.Add(columnValue);

}

if (settings.mass\_Check)

{

DataGridViewCell columnValue = new DataGridViewTextBoxCell();

columnValue.Value = phisics.m;

row.Cells.Add(columnValue);

}

if (settings.q\_Check)

{

DataGridViewCell columnValue = new DataGridViewTextBoxCell();

columnValue.Value = phisics.q(t);

row.Cells.Add(columnValue);

}

if (settings.dq0\_Check)

{

DataGridViewCell columnValue = new DataGridViewTextBoxCell();

columnValue.Value = phisics.dq0;

row.Cells.Add(columnValue);

}

if (settings.c\_Check)

{

DataGridViewCell columnValue = new DataGridViewTextBoxCell();

columnValue.Value = phisics.c;

row.Cells.Add(columnValue);

}

if (settings.w\_Check)

{

DataGridViewCell columnValue = new DataGridViewTextBoxCell();

columnValue.Value = phisics.w;

row.Cells.Add(columnValue);

}

if (settings.E\_Check)

{

DataGridViewCell columnValue = new DataGridViewTextBoxCell();

columnValue.Value = phisics.E;

row.Cells.Add(columnValue);

}

dataTable.Add(row);

}

return dataTable;

}

}

Данный класс имеет всего два публичных метода. Метод CreateTableColumn служит для получения списка названий столбцов в данном графике, который можно настроить вручную. Второй метод CreateTableRow служит уже для заполнения таблицы данными.

## 2.4 Описание метода импорта в Excel

Для сохранения параметром (импорт) был разработан класс «Excels.cs»

public void SaveExcel(DataGridView dataGridView1)

{

SaveFileDialog sfd = new SaveFileDialog();

sfd.Filter = "Excel Documents (\*.xls)|\*.xls";

sfd.FileName = "Excel";

if (sfd.ShowDialog() == DialogResult.OK)

{

// Copy DataGridView results to clipboard

copyAlltoClipboard(dataGridView1);

object misValue = System.Reflection.Missing.Value;

Excel.Application xlexcel = new Excel.Application();

xlexcel.DisplayAlerts = false; // Without this you will get two confirm overwrite prompts

Excel.Workbook xlWorkBook = xlexcel.Workbooks.Add(misValue);

Excel.Worksheet xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get\_Item(1);

// Format column D as text before pasting results, this was required for my data

Excel.Range rng = xlWorkSheet.get\_Range("D:D").Cells;

rng.NumberFormat = "@";

// Paste clipboard results to worksheet range

Excel.Range CR = (Excel.Range)xlWorkSheet.Cells[1, 1];

CR.Select();

xlWorkSheet.PasteSpecial(CR, Type.Missing, Type.Missing, Type.Missing, Type.Missing, Type.Missing, true);

// For some reason column A is always blank in the worksheet. ¯\\_(ツ)\_/¯

// Delete blank column A and select cell A1

Excel.Range delRng = xlWorkSheet.get\_Range("A:A").Cells;

delRng.Delete(Type.Missing);

xlWorkSheet.get\_Range("A1").Select();

// Save the excel file under the captured location from the SaveFileDialog

xlWorkBook.SaveAs(sfd.FileName, Excel.XlFileFormat.xlWorkbookNormal, misValue, misValue, misValue, misValue, Excel.XlSaveAsAccessMode.xlExclusive, misValue, misValue, misValue, misValue, misValue);

xlexcel.DisplayAlerts = true;

xlWorkBook.Close(true, misValue, misValue);

xlexcel.Quit();

releaseObject(xlWorkSheet);

releaseObject(xlWorkBook);

releaseObject(xlexcel);

// Clear Clipboard and DataGridView selection

Clipboard.Clear();

dataGridView1.ClearSelection();

}

}

private void copyAlltoClipboard(DataGridView dataGridView1)

{

dataGridView1.SelectAll();

DataObject dataObj = dataGridView1.GetClipboardContent();

if (dataObj != null)

Clipboard.SetDataObject(dataObj);

}

private void releaseObject(object obj)

{

try

{

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

obj = null;

}

catch (Exception ex)

{

obj = null;

MessageBox.Show("Exception Occurred while releasing object " + ex.ToString());

}

finally

{

GC.Collect();

}

}

Данный проект состоит из одного публичного и двух приватных методов. Главный метод вызывает SaveFileDialog, который указывает путь для сохранения файла. Дальше идет копирование всей таблицы и перенос параметров в Excel файл. В данном методе дальше идет указание с какой ячейке идет заполнение данных. После переноса вызываем метод для очистки и выгрузки данных метод из оперативной памяти, чтобы не загружать систему.

## 2.5 Возможности проекта

Программа начинается с отображения оригинальной заставки, на которой отображание тема курсового проета, а также создатель данного проекта. Затем на экране появляется главное окно программы. Он управляет работой всеми другими модулями в соответствии с запросами пользователя.

Данная форма содержит следующие возможности:

* Передачу данных в Excel;
* Вызов отображения графика;
* Вызов отображение таблицы;
* Вызов формы настройки вводимых значений;
* Вызов формы настройки форм;
* Вызов окна помощи.

Диалог с пользователем поддерживается с помощью главного и выпадающего меню, панели инструментов, кнопок и других интерфейсных элементов.

График служит для отображения информации в графической форме.

# 3 Руководство пользователя

При запуске приложение происходит загрузка системы – это займет пару секунд времени (рисунок 2).

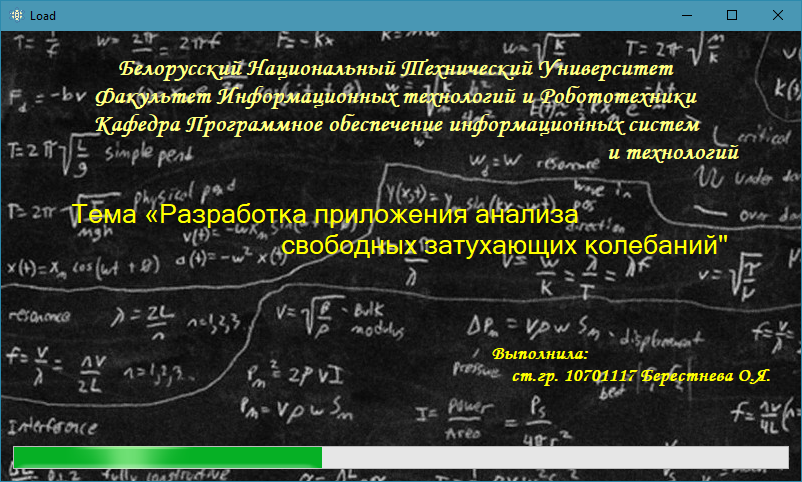


Рисунок 2 – Окно загрузки

После загрузки формы появится следующее окно (рисунок 3).

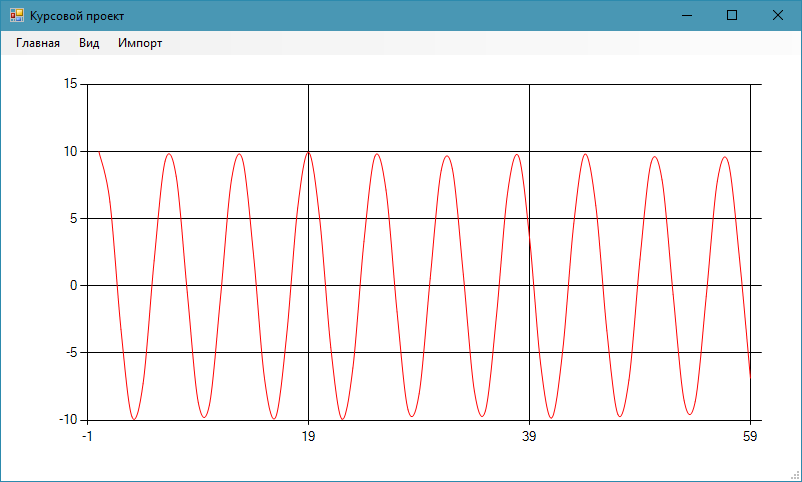


Рисунок 3 – Окно отображения

Для изменения параметров функции, нужно нажать правой кнопкой мыши на график и выбрать в появившемся окне «Параметры» (рисунок 4).

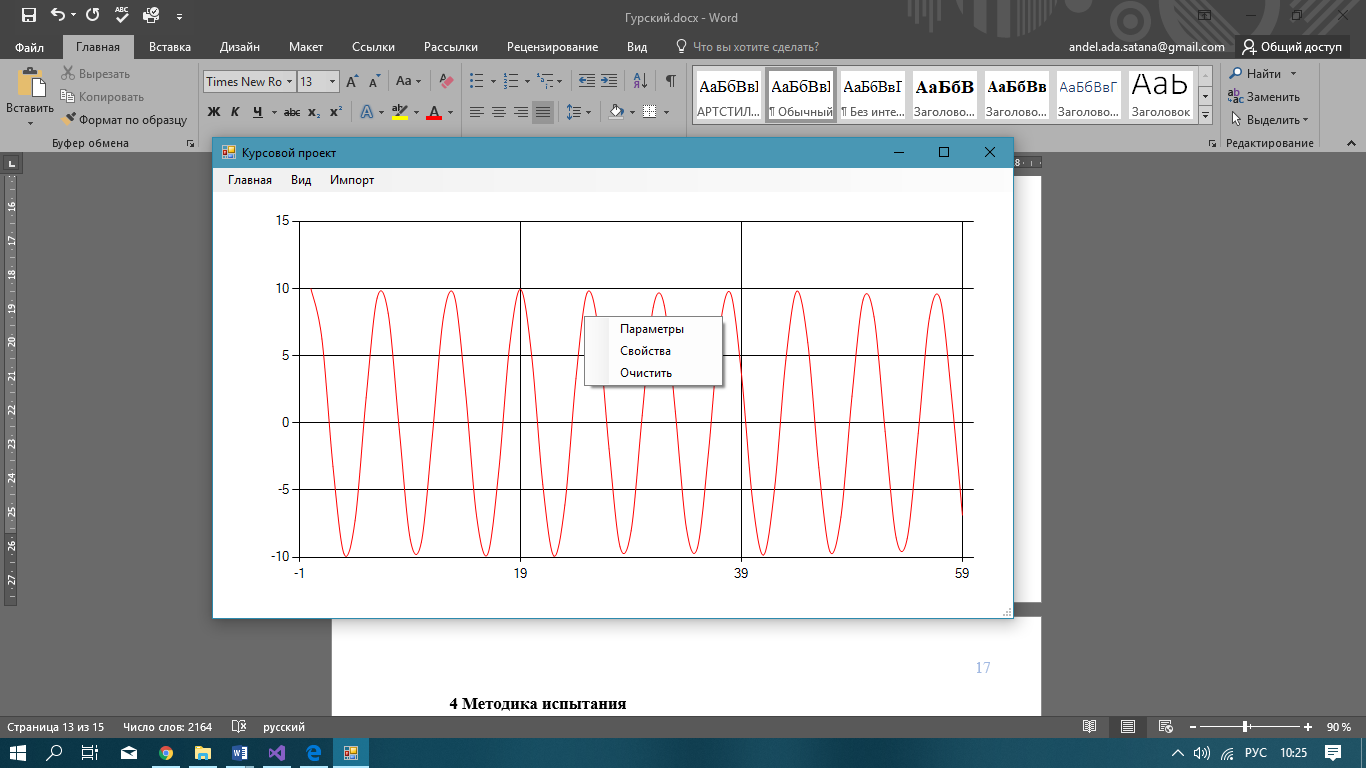


Рисунок 4 – Селектор выбора

Когда нажали на кнопку «Параметры» пользователю откроется окно вводимых параметров функции, а также выбор единиц измерений (рисунок 5)

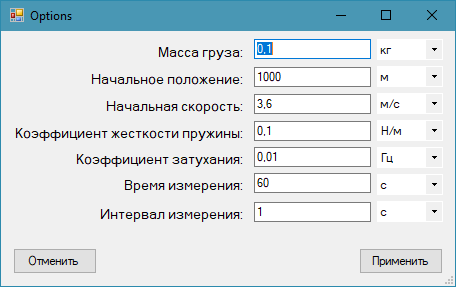


Рисунок 5 – Окно Параметров

На рисунке 4 можно так же выбрать «Свойства». Они служат для настройки графика, толщены линии и других параметров, в том числе и для элемента табличного представления (рисунки 6 и 7).

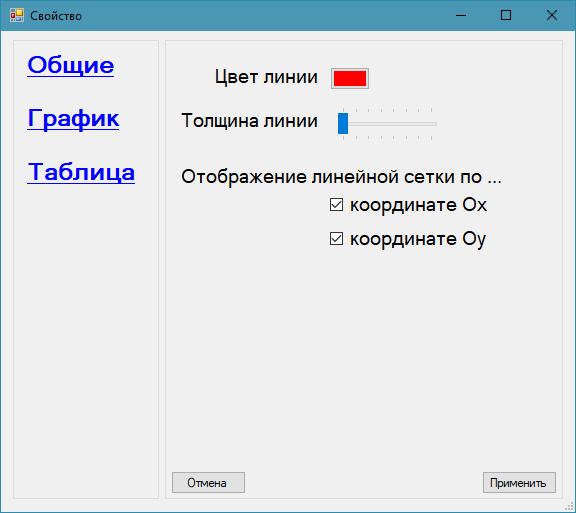


Рисунок 6 – Настройка графика

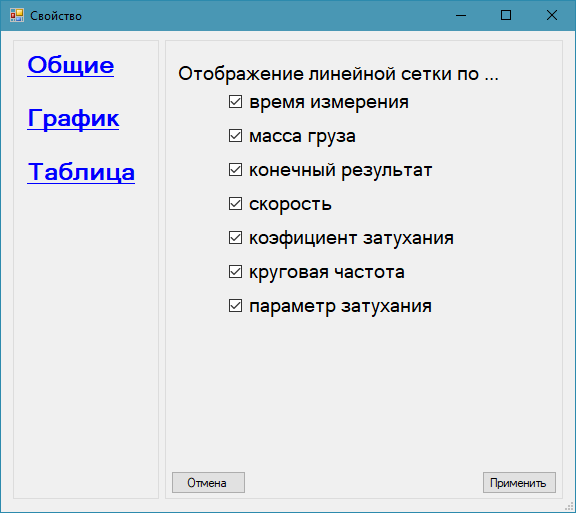


Рисунок 7 – Настройка таблицы

Для изменения представления данных из графика в табличный необходимо нажать на элемент меню «Вид», а там выбрать «Таблица» (рисунок 8)

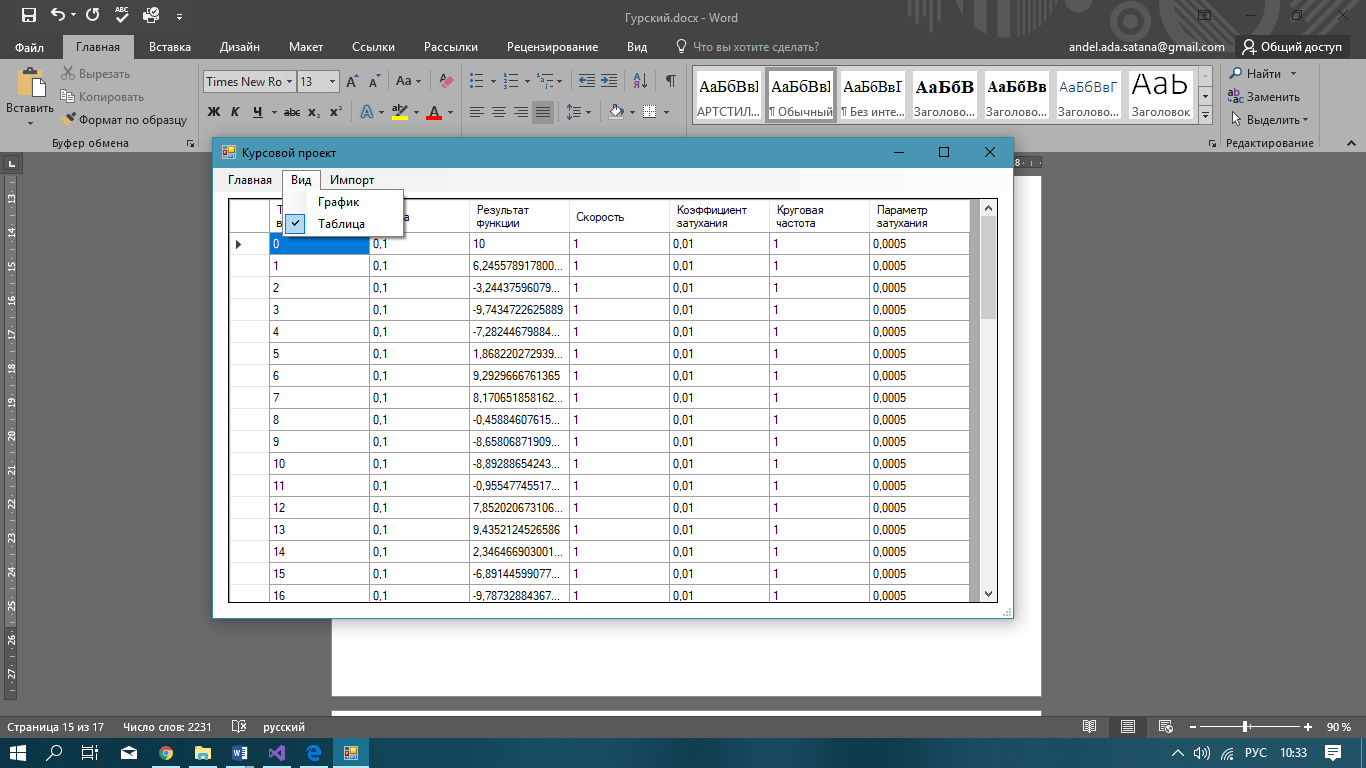


Рисунок 8 – Выбор вида

Так только было выбрана «Таблица» то в рабочей области отобразится таблица (рисунок 9).

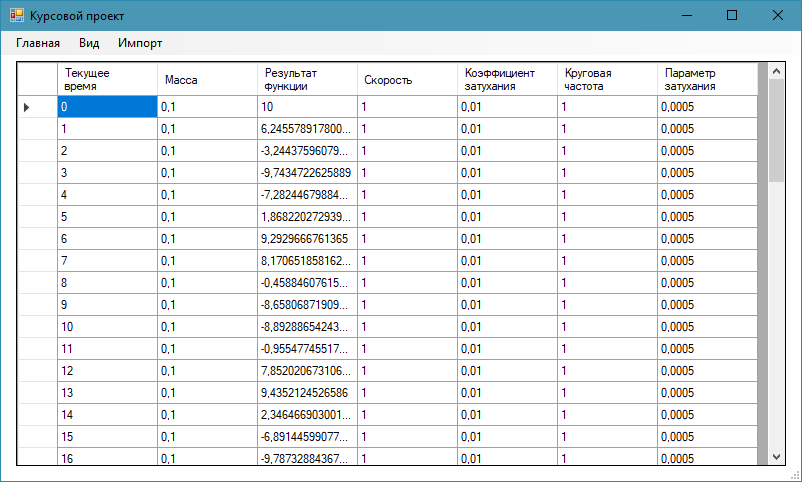


Рисунок 9 – Рабочая зона с таблицей

Чтобы сохранить параметры таблицы в Excel необходимо нажать на «Импорт» и указать путь сохранения данных.

# 4 Методика испытания

Целью проведения испытаний является проверка работоспособности (надежности) программы при различных условиях ее функционирования. Программа должна обеспечивать корректность ввода исходных данных (путем осуществления соответствующих проверок и информирования пользователя о возникших неточностях в работе), а также получение непротиворечивого результата.

Для демонстрации работоспособности программы необходимо провести ряд испытаний с различными начальными условиями.

При начальных условиях, представленных на форме, получаем графический результат как на рисунке 10.

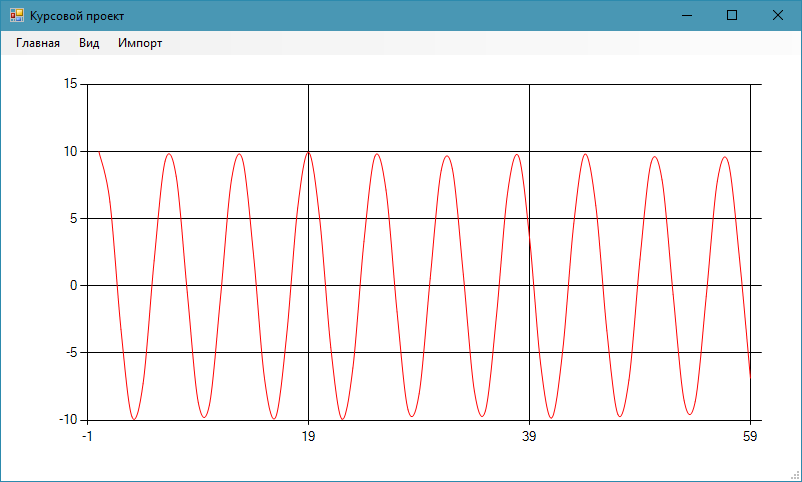


Рисунок 10 – Графический результат начальных параметров

Дальше изменим параметры соответствую введенным параметрам (рисунок 11) получим следующий результат (рисунок 12).

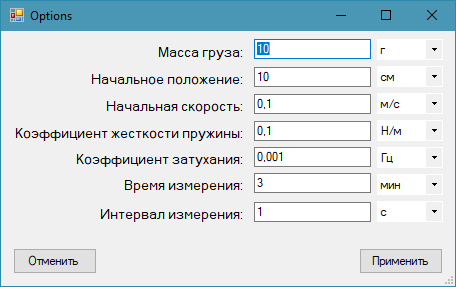


Рисунок 11 – Изменённый параметры

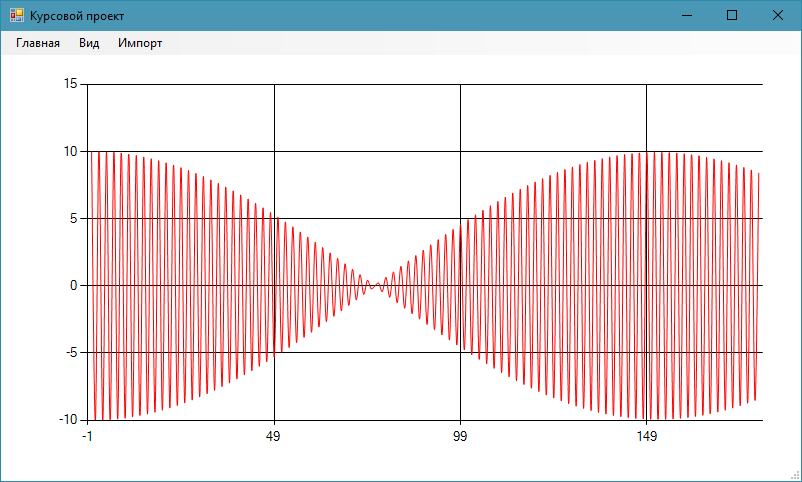


Рисунок 12 – Полученный графический результат

Как видим мы получили еще один тип описанный в разделе 1 «Математическая модель».

Изменим еще раз переменные (рисунок 13) и получим следующий график (рисунок 14).

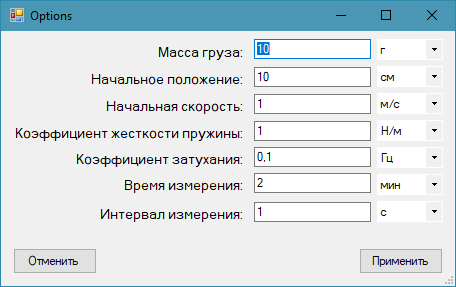


Рисунок 13 – Измененные параметры

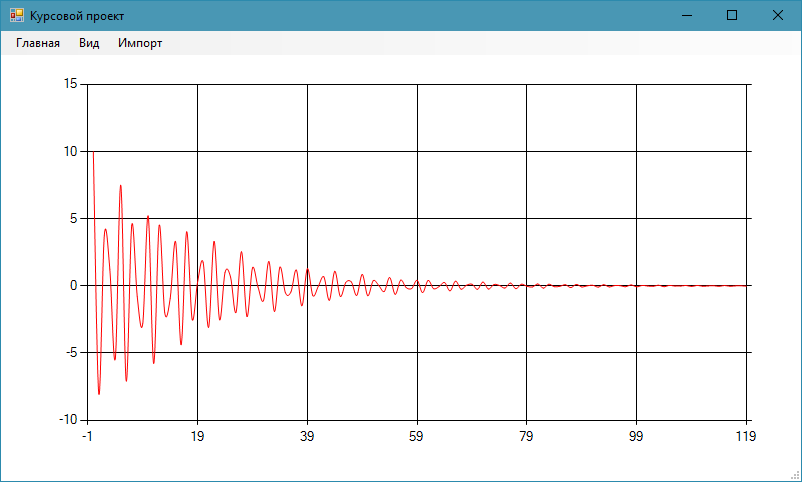


Рисунок 14 – Полученный график

Заметим, что график примял еще один вид, который был описан в первом разделе «Математическая модель»

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы:

* были закреплены знания по курсу «Разработка приложений в визуальных средах»;
* приобретен опыт при разработке объектно-ориентированных программ;
* освоены методы передачи данных между приложениями;
* изучены возможности создания справочной системой высокой степени сложности и различных форматов.

При разработке программы использовался модульный принцип программи-рования. Такая структура программы позволяет в дальнейшем модифицировать от-дельные её части без нарушения работоспособности и потери функциональности приложения.

Программа имеет достаточно ясный и понятый дружелюбный пользовательский интерфейс, обеспечивающий удобство в работе и обеспечивающий получение необходимой пользователю информации с наименьшими затратами времени.

Таким образом, разработанное в ходе выпонения курсовой работы приложение является законченным программным продуктом.

# Список используемой литературы

1. Гринчишин Я.Т. и др. Алгоритмы и программы на Бейсике: Учеб. Пособие для студентов педагогических Институтов по физ.-мат. Спец./Я.Т.Гринчишин, В.И.Ефимов, А.Н.Ломакович.-М.: Просвещение, 1988- 160 с.;
2. Фаронов В.В. Delphi. Программирование на языке высокого уровня: Учебник для вузов – СПб.: Питер, 2005;
3. Архангельский А.Я. Delphi 6: Справочное пособие – Бином, 2001;
4. Федоров Ф. Delphi 5.0 для всех – М.: Компьютер Пресс, 1998.

# Приложение А

Файл Phisics.cs

public class Phisics

{

public Phisics()

{

m = 0.1;

q0 = 10;

dq0 = 1;

k = 0.1;

c = 0.01;

m\_item = 1;

q0\_item = 1;

dq0\_item = 0;

k\_item = 0;

c\_item = 0;

time = 60;

time\_item = 0;

time\_interval = 1;

time\_interval\_item = 0;

}

//Вводимые параметры

public double m { get; set; } // масса

public int m\_item { get; set; }

public double q0 { get; set; } //начальное положение

public int q0\_item { get; set; }

public double dq0 { get; set; } //начальная скорость

public int dq0\_item { get; set; }

public double k { get; set; } // коэфициент жесткости пружины????

public int k\_item { get; set; }

public double c { get; set; } // коэфициент затухания

public int c\_item { get; set; }

public double time { get; set; } // время измерения

public int time\_item { get; set; }

public double time\_interval { get; set; } // интервал измерения

public int time\_interval\_item { get; set; }

//Неизвестные

private double n //

{

get

{

return c / (2 \* m);

}

}

public double w // круговая частота свободного колебания

{

get

{

return Math.Sqrt(k / m);

}

}

public double E // параметр затухания

{

get

{

return c / 2 \* m \* w;

}

}

private double Wd // частота колебания с затуханием или чаcтота демпфированных колебаний

{

get

{

return w \* Math.Sqrt(1 - Math.Pow(E, 2));

}

}

private double WV

{

get

{

return w \* Math.Sqrt(Math.Pow(E, 2) - 1);

}

}

public double q(double t)

{

double result = 0;

if (n == w) // критическое затухание

result = (q0 \* (1 + w \* t) + dq0 \* t) \* Math.Pow(Math.E, -1 \* w \* t);

if (n < w) // недодемпфированные системы ( с затуханием ниже критического)

result = Math.Pow(Math.E, -1 \* w \* t \* E) \* (((dq0 + q0 \* E \* w) / Wd) \* Math.Sin(Wd \* t) + q0 \* Math.Cos(Wd \* t));

if (n > w) // Передемпфированные системы ( с затуханием больше критического)

result = Math.Pow(Math.E, -1 \* w \* t \* E) \* (((dq0 + q0 \* E \* w) / WV) \* sh(WV \* t) + q0 \* ch(WV \* t));

return result;

}

private double sh(double x)

{

return (Math.Pow(Math.E, x) - Math.Pow(Math.E, -1 \* x)) / 2;

}

private double ch(double x)

{

return (Math.Pow(Math.E, x) + Math.Pow(Math.E, -1 \* x)) / 2;

}

//перевод в единицы измерения СИ

public double CI\_mass(double parents, int position )

{

if(position == 0)

{

parents = parents / 1000;

}

return parents;

}

public double CI\_q0(double parents, int position)

{

if (position == 1)

{

parents = parents / 100;

}

return parents;

}

public double CI\_dq0(double parents, int position)

{

if (position == 0)

{

parents = parents \* 1000 / 3600;

}

return parents;

}

public double CI\_time(double parents, int position)

{

if (position == 1)

{

parents = parents \* 60;

}

return parents;

}

public double CI\_timeInterval(double parents, int position)

{

if (position == 1)

{

parents = parents \* 60;

}

return parents;

}

//перевод из единиц измерения СИ

public double NoCI\_mass(int position)

{

if (position == 0)

{

m = m \* 1000;

}

return m;

}

public double NoCI\_q0(int position)

{

if (position == 1)

{

q0 = q0 \* 100;

}

return q0;

}

public double NoCI\_dq0(int position)

{

if (position == 0)

{

dq0 = dq0 / 1000 \* 3600;

}

return dq0;

}

public double NoCI\_time(int position)

{

if (position == 1)

{

time = time / 60;

}

return time;

}

public double NoCI\_timeInterval(int position)

{

if (position == 1)

{

time\_interval = time\_interval / 60;

}

return time\_interval;

}

}

Файл Points.cs

public class Points

{

List<Points> listPoints = new List<Points>();

public double X { get; set; }

public double Y { get; set; }

public Points() { }

private Points(double x, double y)

{

this.X = x;

this.Y = y;

}

private void AddPoints(Phisics phisics)

{

for (double time = 0; time < phisics.time; time += phisics.time\_interval)

listPoints.Add(new Points(time, phisics.q(time)));

}

public List<Points> PointsList(Phisics phisics)

{

AddPoints(phisics);

return listPoints;

}

}

Файл Excels.cs

public class Excels

{

public void SaveExcel(DataGridView dataGridView1)

{

SaveFileDialog sfd = new SaveFileDialog();

sfd.Filter = "Excel Documents (\*.xls)|\*.xls";

sfd.FileName = "Excel";

if (sfd.ShowDialog() == DialogResult.OK)

{

// Copy DataGridView results to clipboard

copyAlltoClipboard(dataGridView1);

object misValue = System.Reflection.Missing.Value;

Excel.Application xlexcel = new Excel.Application();

xlexcel.DisplayAlerts = false; // Without this you will get two confirm overwrite prompts

Excel.Workbook xlWorkBook = xlexcel.Workbooks.Add(misValue);

Excel.Worksheet xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get\_Item(1);

// Format column D as text before pasting results, this was required for my data

Excel.Range rng = xlWorkSheet.get\_Range("D:D").Cells;

rng.NumberFormat = "@";

// Paste clipboard results to worksheet range

Excel.Range CR = (Excel.Range)xlWorkSheet.Cells[1, 1];

CR.Select();

xlWorkSheet.PasteSpecial(CR, Type.Missing, Type.Missing, Type.Missing, Type.Missing, Type.Missing, true);

// For some reason column A is always blank in the worksheet. ¯\\_(ツ)\_/¯

// Delete blank column A and select cell A1

Excel.Range delRng = xlWorkSheet.get\_Range("A:A").Cells;

delRng.Delete(Type.Missing);

xlWorkSheet.get\_Range("A1").Select();

// Save the excel file under the captured location from the SaveFileDialog

xlWorkBook.SaveAs(sfd.FileName, Excel.XlFileFormat.xlWorkbookNormal, misValue, misValue, misValue, misValue, Excel.XlSaveAsAccessMode.xlExclusive, misValue, misValue, misValue, misValue, misValue);

xlexcel.DisplayAlerts = true;

xlWorkBook.Close(true, misValue, misValue);

xlexcel.Quit();

releaseObject(xlWorkSheet);

releaseObject(xlWorkBook);

releaseObject(xlexcel);

// Clear Clipboard and DataGridView selection

Clipboard.Clear();

dataGridView1.ClearSelection();

}

}

private void copyAlltoClipboard(DataGridView dataGridView1)

{

dataGridView1.SelectAll();

DataObject dataObj = dataGridView1.GetClipboardContent();

if (dataObj != null)

Clipboard.SetDataObject(dataObj);

}

private void releaseObject(object obj)

{

try

{

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

obj = null;

}

catch (Exception ex)

{

obj = null;

MessageBox.Show("Exception Occurred while releasing object " + ex.ToString());

}

finally

{

GC.Collect();

}

}

}

Файл Settings.cs

public class Settings

{

public Color colorLineGraf { get; set; }

public int lineThickness { get; set; }

public bool AsixX { get; set; }

public bool AsixY { get; set; }

public Settings()

{

colorLineGraf = Color.Red;

lineThickness = 1;

AsixX = true;

AsixY = true;

time\_Check = true;

mass\_Check = true;

q\_Check = true;

dq0\_Check = true;

c\_Check = true;

w\_Check = true;

E\_Check = true;

}

public bool time\_Check { get; set; }

public bool mass\_Check { get; set; }

public bool q\_Check { get; set; }

public bool dq0\_Check { get; set; }

public bool c\_Check { get; set; }

public bool w\_Check { get; set; }

public bool E\_Check { get; set; }

}

Файл Tables.cs

class Tables

{

public List<DataGridViewTextBoxColumn> CreateTableColumn(Settings settings)

{

List<DataGridViewTextBoxColumn> column = new List<DataGridViewTextBoxColumn>();

if (settings.time\_Check)

{

DataGridViewTextBoxColumn col0 = new DataGridViewTextBoxColumn();

col0.HeaderText = "Текущее время";

col0.Name = "ID";

column.Add(col0);

}

if (settings.mass\_Check)

{

DataGridViewTextBoxColumn col0 = new DataGridViewTextBoxColumn();

col0.HeaderText = "Масса";

col0.Name = "mass";

column.Add(col0);

}

if (settings.q\_Check)

{

DataGridViewTextBoxColumn col0 = new DataGridViewTextBoxColumn();

col0.HeaderText = "Результат функции";

col0.Name = "func";

column.Add(col0);

}

if (settings.dq0\_Check)

{

DataGridViewTextBoxColumn col0 = new DataGridViewTextBoxColumn();

col0.HeaderText = "Скорость";

col0.Name = "dq0";

column.Add(col0);

}

if (settings.c\_Check)

{

DataGridViewTextBoxColumn col0 = new DataGridViewTextBoxColumn();

col0.HeaderText = "Коэффициент затухания";

col0.Name = "c";

column.Add(col0);

}

if (settings.w\_Check)

{

DataGridViewTextBoxColumn col0 = new DataGridViewTextBoxColumn();

col0.HeaderText = "Круговая частота";

col0.Name = "w";

column.Add(col0);

}

if (settings.E\_Check)

{

DataGridViewTextBoxColumn col0 = new DataGridViewTextBoxColumn();

col0.HeaderText = "Параметр затухания";

col0.Name = "e";

column.Add(col0);

}

return column;

}

public List<DataGridViewRow> CreateTableRow(Phisics phisics, Settings settings)

{

List<DataGridViewRow> dataTable = new List<DataGridViewRow>();

for (double t = 0; t < phisics.time; t += phisics.time\_interval)

{

DataGridViewRow row = new DataGridViewRow();

if (settings.time\_Check)

{

DataGridViewCell columnValue = new DataGridViewTextBoxCell();

columnValue.Value = t;

row.Cells.Add(columnValue);

}

if (settings.mass\_Check)

{

DataGridViewCell columnValue = new DataGridViewTextBoxCell();

columnValue.Value = phisics.m;

row.Cells.Add(columnValue);

}

if (settings.q\_Check)

{

DataGridViewCell columnValue = new DataGridViewTextBoxCell();

columnValue.Value = phisics.q(t);

row.Cells.Add(columnValue);

}

if (settings.dq0\_Check)

{

DataGridViewCell columnValue = new DataGridViewTextBoxCell();

columnValue.Value = phisics.dq0;

row.Cells.Add(columnValue);

}

if (settings.c\_Check)

{

DataGridViewCell columnValue = new DataGridViewTextBoxCell();

columnValue.Value = phisics.c;

row.Cells.Add(columnValue);

}

if (settings.w\_Check)

{

DataGridViewCell columnValue = new DataGridViewTextBoxCell();

columnValue.Value = phisics.w;

row.Cells.Add(columnValue);

}

if (settings.E\_Check)

{

DataGridViewCell columnValue = new DataGridViewTextBoxCell();

columnValue.Value = phisics.E;

row.Cells.Add(columnValue);

}

dataTable.Add(row);

}

return dataTable;

}

}

Файл Form1.cs

public partial class Form1 : Form

{

Phisics phisics;

Points points;

List<Points> listPoints;

Settings settings;

Tables tables = new Tables();

public Form1()

{

InitializeComponent();

panelTable.Location = panelGraf.Location = new Point(12, 27);

panelTable.Size = panelGraf.Size = new Size(776, 411);

dataGridView1.Size = chart1.Size = new Size(770,405);

phisics = new Phisics();

listPoints = new List<Points>();

points = new Points();

settings = new Settings();

listPoints = points.PointsList(phisics);

SetSetting();

GrafMenu.PerformClick();

Timer timer = new Timer();

timer.Interval += 1000;

timer.Tick += new EventHandler(Point);

timer.Start();

}

private void chart1\_MouseClick(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Right)

contextMenuStrip1.Show(MousePosition, ToolStripDropDownDirection.Right);

}

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

{

chart1.Series[0].Points.Clear();

}

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

{

Setting form = new Setting(settings, 1);

form.ShowDialog();

settings = form.settings;

SetSetting();

}

private void Point(object sender, EventArgs e)

{

chart1.Series[0].Points.Clear();

foreach (Points point in listPoints)

{

chart1.Series[0].Points.AddXY(point.X, point.Y);

}

}

private void Table(object sender, EventArgs e)

{

}

private void SetSetting()

{

chart1.Series[0].Color = settings.colorLineGraf;

chart1.Series[0].BorderWidth = settings.lineThickness;

chart1.ChartAreas[0].AxisX.MajorGrid.Enabled = settings.AsixX;

chart1.ChartAreas[0].AxisY.MajorGrid.Enabled = settings.AsixY;

}

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

{

listPoints.Clear();

Options form = new Options(phisics);

form.ShowDialog();

phisics = form.phisics;

listPoints = points.PointsList(phisics);

}

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

{

Close();

}

private void settingMenu\_Click\_1(object sender, EventArgs e)

{

Setting form = new Setting(settings);

form.ShowDialog();

settings = form.settings;

SetSetting();

}

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

{

GrafMenu.Checked = true;

TableMenu.Checked = false;

panelGraf.Visible = true;

panelTable.Visible = false;

}

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

{

GrafMenu.Checked = false;

TableMenu.Checked = true;

panelGraf.Visible = false;

panelTable.Visible = true;

dataGridView1.Rows.Clear();

dataGridView1.Columns.Clear();

List<DataGridViewTextBoxColumn> dataTableColumb = tables.CreateTableColumn(settings);

List<DataGridViewRow> dataTableRow = tables.CreateTableRow(phisics, settings);

foreach (DataGridViewTextBoxColumn data in dataTableColumb)

{

dataGridView1.Columns.Add(data);

}

foreach (DataGridViewRow data in dataTableRow)

{

dataGridView1.Rows.Add(data);

}

}

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

{

//if(saveFileDialog1.ShowDialog() == DialogResult.OK)

//{

new Excels().SaveExcel(dataGridView1);

//}

}

}