ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по дисциплине «Вычислительная математика»

на тему «Численное дифференцирование»

# Введение

В современных науке и технике важную роль играет математическое моделирование, заменяющее эксперименты с реальными объектами экспериментами с их математическими моделями. Но для практических задач довольно редко удается найти аналитическое решение уравнений, составляющих математическую модель явления. Поэтому приходится применять численные методы.

Численное решение прикладных задач всегда интересовало математиков. Резкий скачок в развития вычислительной техники положил начало бурному развитию численных методов.

Численные методы представляют собой набор алгоритмов, позволяющих получать приближенное численное решение поставленных математических задач. Как и любой другой численный метод, эти методы позволяют с заданной точностью получить нужные результаты, используя заданные алгоритмы, не прибегая к выполнению аналитических преобразований над входными данными. Одной из подзадач численных методов, является численное дифференцирование. Численное дифференцирование облегчает работу в случае, если выполнение аналитических преобразований достаточно трудоёмко или же исходные данные, то есть функция, подлежащая дифференцированию, представляет собой результаты проведения экспериментов. Кроме того, численное дифференцирование широко используется при разработке численных методов решения многих задач (решение дифференциальных уравнений, поиск решений не линейных уравнений, поиск точек экстремума функции и др.).

1. Обзорметодовчисленногодифференцирования

.1 Вычисление производной, используя простейшие формулы

Предположим, что функция f дифференцируема в окрестности точки х достаточное количество раз[1]. Исходя из определения производной:



можно получить две простейшие приближенные формулы:





где h - малый параметр (шаг).

Эти формулы часто называют правой и левой разностными производными.

Оценка погрешностей данных формулы производится по следующей формуле:



где



где  - точка, принадлежащая промежутку ).

Таким образом, формулы левых и правых разностных производных имеют первый порядок точности.

Для вычисления второй производной данным методом применяется следующая формула:



Данную величину также называют второй разностной производной.

Для вычисления погрешности, воспользуемся соответствующим разложением по формуле Тейлора:



отсюда получаем:



Тогда для оценки погрешности можно использовать следующее неравенство:



Таким образом вторая разностная производная имеет второй порядок точности.

Приведенные формулы численного дифференцирования имеют простую геометрическую интерпретацию. На рисунке 1.1 a) изображен график функции и отмечены точки N-, N0 и N+, с координатами (x-h, f(x-h)), (x,f(x)) и (x+h,f(x+h)) соответственно.



Рисунок 1.1 - Графики функций

Производная f’(x) равна tg() относительно оси Ox в точке N0. Тангенс угла наклона прямых N- N0 и N+ N0, близок к значению производной в точке х. Однако можно заметить, что тангенс прямой N- N+, которая изображена на рисунке 1.1 б), более близок к искомому значению.

Логично предположить, что можно использовать тангенс прямой N- N+ для более точного нахождения производной в указанной точке. Получившаяся формула выглядит следующим образом:



Данную формулу также называют центральной разностной производной.

Для определения погрешности используют следующую формулу:



где



Формула имеет вторую степень точности.

Таким образом можно получить формулы любой точности, однако для этого нужно знать значения функции в большем количестве точек.

Например, формула, имеющая четвертый порядок, точности имеет следующий вид:



Данный способ удобен в тех случаях, когда точка лежит на заданных узлах, но вычислить производную точки, не принадлежащей узлам, не предоставляется возможным.

.2 Численное дифференцирование, основанное на интерполяции алгебраическими многочленами

Предположим, что f(x) в окрестности точки x аппроксимируется некоторой другой функцией g(x), и производная g’(x) легко вычисляется[2]. Тогда:



Пусть Pn(x) - интерполяционный многочлен n-ой степени с узлами интерполяции x0 < x1<x2<…<xn. В таком случае:



Для получения интерполяционного многочлена применяется аппроксимация многочленом Лагранжа.

Для получения полинома Лагранжа используется следующая формула:



где l0,l1…ln - многочлены, зависящие от x0,x1…xn.

Неизвестные многочлены находятся по формуле:



Таким образом можно записать компактно формулу полинома Лагранжа:



Если использовать данную формулу для получения полинома первой степени, получится следующее выражение:



Из данного выражения можно получить другой алгоритм для интерполяции.

В общем случае полином имеет следующую форму:



Используя это выражение, составим систему линейных алгебраических уравнений, следующего вида:



Решением данной системы будут коэффициенты при соответствующих степенях полинома.

.3 Обусловленность численного дифференцирования

Несмотря на внешнюю простоту формул численного дифференцирования, их применение требует особой осторожности[3]. Отметим, что используемые при численном дифференцировании значения функции не в узловых точках содержат ошибки. Поэтому к погрешности формул численного дифференцирования добавляется не устранимая погрешность, вызванная погрешностями вычисления функции. Для того, что бы погрешность аппроксимации была достаточно малой, требуется очень маленькое значение шага h. Однако, при малых шагах формулы численного дифференцирования становятся плохо обусловленными и результат их применения может быть полностью искажен неустранимой ошибкой. Действительная причина этого явления лежит не в несовершенстве предложенных методов вычисления производных, а в некорректности самой операции дифференцирования приближенно заданной функции.

В качестве примера приведем формулу правой разностной производной.

Полная погрешность



представляет собой сумму погрешностей аппроксимации



и не устранимой погрешности



Пусть  = | f(x) - f\*(x)| - верхняя граница абсолютной погрешности. Тогда погрешность оценивается следующим образом:



Эта оценка означает, что чувствительность формулы к погрешностям входных данных характеризуется абсолютным числом обусловленности



Так как  при , то формула дифференцирования становится очень плохо обусловленной. Поэтому, несмотря на то, что погрешность аппроксимации стремится к нулю, следует ожидать, что полная погрешность будет неограниченно возрастать. Во всяком случае так себя ведет верхняя граница полной погрешности



Таким образом, при использовании формул для вычисления производной функции f, заданном с погрешностью, следует обратить внимание на выбор шага. Однако даже при оптимальном выборе шага полная погрешность окажется величиной, пропорциональной лишь  (для формулы правой разностной производной).

Формулы для вычисления производных высших порядков обладают еще большей чувствительностью к ошибкам задания функции. Поэтому значения производных высокого порядка с помощью данных методов, могут быть очень не точными.

Производная находит широкое приложение в физике для нахождения скорости по известной функции координаты от времени, ускорения по известной функции скорости от времени; для нахождения наибольших и наименьших величин.

Производная является важнейшим инструментом экономического анализа, позволяющим углубить геометрический и математический смысл экономических понятий, а также выразить ряд экономических законов с помощью математических формул.

Наиболее актуально использование производной в предельном анализе, то есть при исследовании предельных величин (предельные издержки, предельная выручка, предельная производительность труда или других факторов производства и т. д.).

Производная применяется в экономической теории. Многие, в том числе базовые, законы теории производства и потребления, спроса и предложения оказываются прямыми следствиями математических теорем

Если сравнивать описанные способы численного дифференцирования, то можно заметить, что методы левой, центральной и правой разностей довольно просто реализуются, на их выполнение требуется меньше ресурсов и времени. Однако, дифференцирование с помощью замены функции интерполяционным многочленом позволяет узнать значении производной в точках, отличных от узловых, но этот метод имеет большие затраты времени. К тому же, при маленьких значения шага h и увеличения кол-ва данных шагов значительно повышается погрешность, хотя и находится многочлен, который более приближен к функции.

2. Математическаяпостановказадачичисленногодифференцирования

.1 Разностные формулы численного дифференцирования

Для нахождения производной данным способом необходимо установить, к какому узлу принадлежит искомая точка. Возможные варианты: искомый узел крайний правый, искомый узел крайний левый, искомый узел не является крайними. Дальше, в зависимости от узла, необходимо применить определенную формулу.

**Входные данные:**

Множество точек Mat = такое, что

а) .

б) 

Точка, в которой нужно найти производную: .

**Выходные данные:**

Значение производной в указанной точке: 

**Аналитическое решение:**







Использования конкретной формулы зависит от того, где находится точка, производную которой нужно вычислить. Если это точка крайняя слева, то используется первая формула, если точка крайняя справа, то используется вторая формула, в остальных случаях используется третья формула.

**Алгоритм дифференцирования методом конечных разностей:**

1) Найти .

2) PointInd = i.

*3)* 

) Если i = 0, тогда .

) Если i = n, тогда 

) Если  тогда .

**Графическое представление алгоритма:**

На рисунке 2.1 изображена блок-схема алгоритма дифференцирования с использованием разностных формул.



Рисунок 2.1 - Блок-схема функции вычисления производной методом разностей

2.2 Численное дифференцирование, с использованием интерполяции

Для решения задачи данными методом необходимо найти интерполяционный многочлен Лагранжа. Для его поиска составляется специальная система уравнений, решением которой являются коэффициенты при соответствующих степенях многочлена. В дальнейшем данный многочлен дифференцируется. Подставляя заданную точку в многочлен, получившийся после дифференцирования, получаем примерное значение производной в данной точке.

**Входные данные:**

Точка, в которой нужно найти производную: .

Множество точек Mat =такое, что

.

**Выходные данные:**

Значение производной в указанной точке: 

**Аналитическое решение:**







**Алгоритм дифференцирования с использованием полинома Лагранжа:**

1) Для матрицы  для .

2)  для 

) Решаем систему уравнений , относительно .

*4)* 

) Заполнить массив . Для делать 

) Для  делать 

7) 

**Графическое представление алгоритма:**

На рисунке 2.2 изображена блок-схема алгоритма дифференцирования с использованием полинома Лагранжа.



Рисунок 2.2 - Блок-схема алгоритма решения поставленной задачи

. Тестирование разработанной программы

На рисунке 3.1 представлена работа программы на одном из тестов.



Рисунок 3.1 - Работа программы во время второго теста

Тестирование программы производилось для функций, значение производной которых можно вычислить аналитически.

Тестирования производилось для следующих функций:









Результаты тестов представлены в таблицах 3.1, 3.2, 3.3, и 3.4.

Первые два столбца таблицы - это точка х и значение производной в данной точке, полученное аналитически, соответственно. Следующие два столбца - значение производной, вычисленных с помощью формул численного дифференцирования (

). Следующие два столбца - погрешность вычисленных значений. Последние два столбца - среднее время работы методов, измеряемое в тактах процессора.

Результаты теста, при котором использовалась функция представлены в таблице 3.1. Аналитически полученная формула производной: 

Таблица 3.1 - Первый тест программы

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| x |  |  |  |  |  |  |  |
| 0 | 1 | 0,90032 | 1,16401 | 0,09968 | -0,16401 | 4925 | 18100  |
| 0,78539 | 0,70710 | 0,63663 | 0,63663 | 0,07047 | 0,07047 | 5660 | 18400 |
| 1,57079 | 0 | 0,37294 | 0,10925 | -0,37294 | -0,10925 | 5140 | 17143 |

По результату первого теста можно заметить, что точность методов колеблется в различных точках. Формулы разностных производных работают быстрее полиномиальной интерполяции примерно в 3.4 раза быстрее.

Второй тест был проведен с использованием функции  Производная данной функции задается следующей формулой: Результаты теста представлены в таблице 3.2. В данном тесте функция задавалась четырьмя точками.

Таблица 3.2 - Второй тест программы

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| x |  |  |  |  |  |  |  |
| -1 | -2 | -1 | -2 | -1 | 0 | 5350 | 21400 |
| 0 | 0 | 0 | 0 | 0 | 0 | 4910 | 20000 |
| 1 | 2 | 2 | 0 | 2 | 0 | 4800 | 22650 |
| 2 | 4 | 3 | 1 | 4 | 0 | 4900 | 20550 |

В данном тесте интересно то, что с помощью полиномиальной интерполяции функция была найдена точно, поэтому погрешность производной равна 0. При других опытах было замечено, что, если функция изначально была полиномом, то этот полином часто находится точно, или же он получается близким к исходному. При использовании разностных формул в крайних точках была получена довольно большая погрешность, а вот в остальных точках погрешность отсутствует. При данных теста было получено, что скорость разностных формул быстрее примерно в 4 раза.

При третьем тесте была использована функция  Ее производная совпадает с самой функцией. Было использовано 5 точек. Результаты тестирования представлены в таблице 3.3.

Таблица 3.3 - Третий тест программы

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| x |  |  |  |  |  |  |  |
| 0,2 | 1,2214 | 1,3521 | 1,22092 | -0,1307 | 0,00048 | 5470 | 24550 |
| 0,4 | 1,49182 | 1,50178 | 1,49192 | -0,01005 | -0,0001 | 4700 | 25200 |
| 0,6 | 1,82211 | 1,8343 | 1,82203 | -0,01219 | 0,00008 | 5150 | 24000 |
| 0,8 | 2,22554 | 2,24043 | 2,22574 | -0,01489 | -0,0002 | 5100 | 24000 |
| 1 | 2,71828 | 2,4637 | 2,71755 | 0,25458 | 0,00073 | 5300 | 23800 |

В третьем тесте погрешность метода, использующего полиномиальную интерполяцию, во много раз меньше, чем разностных формул. В данном тесте метод разностной производной быстрее в 4.7 раза. последнем тесте была использована следующая функция: . Производная этой функции выглядит так: В программе функция была задана шестью точками. Результаты теста в таблице 3.4.

Таблица 3.4 -Четвертый тест программы

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| x |  |  |  |  |  |  |  |
| -1 | -0,36787 | -0,43248 | -0,17795 | 0,06461 | -0,18992 | 5000 | 28900 |
| -0,5 | -0,45489 | -0,36787 | -0,49623 | -0,08702 | 0,04134 | 5200 | 27800 |
| 0 | 0 | 0,26055 | 0,02263 | -0,26055 | -0,02263 | 4750 | 29300 |
| 0,5 | 2,0609 | 2,71828 | 2,03589 | -0,65738 | 0,02501 | 5550 | 30330 |
| 1 | 8,15484 | 9,67162 | 8,21011 | -1,51678 | -0,05527 | 5050 | 32200 |
| 1,5 | 23,52886 | 14,73104 | 23,22028 | 8,79782 | 0,30858 | 5900 | 33900 |

Последний тест отличался от предыдущих тем, что функция была сложной, а не элементарной. В данном тести по мере роста функции погрешность формулы разностей увеличивалась на много быстрее. Но так же данная формула оказалась быстрее уже в 5.8 раза.

На рисунке 3.2 показан график зависимости отношения  от количества заданных точек.



Рисунок 3.2 - График зависимости  от количества точек

Из графика видно прямую зависимость  от количества заданных точек.

На основании данных тестов можно сделать вывод, что разностные формулы численного дифференцирования в большинстве случаев менее точны, чем использование интерполяционного полинома Лагранжа, однако они работают значительно быстрее.

Также были проведены тесты, когда были введены неверные входные данные (различные значения функции в одинаковых точках, данные, не соответствующие формату и др.). Во время этих тестов ошибок, приводящих к не правильному решению или экстренному завершению программы, обнаружено не было.

Заключение

численныйдифференцирование многочлен интерполяция

В результате проделанной работы были изучены следующие методы численного дифференцирования: формулы разностных производных, дифференцирования с помощью интерполяционного многочлена Лагранжа.

У каждого метода были выявлены сильные и слабые стороны.

Сильной стороной формул разностных производных является быстрота их выполнения. Однако данные формулы не могут рассчитывать производные в точках, отличных от узловых. Также, на основанию проведенных опытов было установлено, что данные формулы обычно имеют меньшую точность.

Дифференцирование с использование интерполяционного многочлена Лагранжа позволяет находить значение производной в точках, отличных от узловых. Также имеет в большинстве случаев большую точность. Если известно, что функция должна представлять полином, то погрешность метода стремится к нулю. Однако, данный метод требует больших вычислительных затрат, т.к. требуется решать систему уравнений, возводить числа в степень.

Был составлен алгоритм решения задачи различными способами.

На основании алгоритмов была составлена программа, реализующая данные методы.

В ходе тестов программы не было обнаружено критических ошибок.

Тестирование программы на различных данных показало, что формулы численного дифференцирования могут использоваться в практических целях. Однако их нужно использовать осторожно, т.к. каждый из методов в определенных точках может иметь очень большую погрешность.

Список литературы

1 А.А.Амосов, Ю.А.Дубинский, Вычислительные методы для инженеров. - М.: Высш.шк., 2014. - с.544

2 David Kincaid and Ward Cheney, Numerical analysis: mathematics of scientific computing. - Wadsworth, Inc., Belmont, California, 2011. - с.701

3 Н. С. Бахвалов, Численные методы. - М.: Наука, 2009. - с.190

Приложения

Приложение А

Текст программы

Модуль Form1.cs:

using System;System.Collections.Generic;System.ComponentModel;System.Data;System.Drawing;System.Linq;System.Text;System.Windows.Forms;System.Diagnostics;Kursavik\_sem\_3

{partial class Form1 : Form

{Form1()

{();

}dif;Razn;Interp;form2;void Form1\_Load(object sender, EventArgs e)

{

//задачем начальные значения программы.Text = "The programm";

listBox1.Items.Add("0,2 1,2214");.Items.Add("0,4 1,49182");.Items.Add("0,6 1,82211");.Items.Add("0,8 2,22554");.Items.Add("1,0 2,71828");= new RaznDiffer(label8);= new InterDiffer(label8);= Razn;.SelectedItem = 0;.SelectedIndex = 0;.ResetText();= new Form2(this);.Invalidate();

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

{.ResetText();x, y;

try

{

//если значения нормально конвернтируются

//то добавляем их к другим точкми

x = Convert.ToDecimal(textBox1.Text);= Convert.ToDecimal(textBox2.Text);strX, strY;= Convert.ToString(x);= Convert.ToString(y);.Items.Add(x+" "+y);

}(FormatException)

{.Show("Invalid data!\nPlease, enter other data.", "Error converting");

}

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

{

{.SelectedObjectCollection col = listBox1.SelectedItems;(col.Count!=0)

{.Items.Remove(col[0]);

}

}(ArgumentOutOfRangeException)

{.Show("None selected", "Error");

}

}void textBox1\_KeyDown(object sender, KeyEventArgs e)

{(e.KeyValue == 13) button1\_Click(this, e);

}void StrDiv(string str, out decimal x, out decimal y)

{

//разбираем строку на состовляющие числа.

string strF="";i = 0;(str[i] != ' ') strF += str[i++];= Convert.ToDecimal(strF);= "";(i < str.Length) strF += str[i++];= Convert.ToDecimal(strF);

}void button3\_Click(object sender, EventArgs e)

{x1;

{= Convert.ToDecimal(textBox3.Text);

}(FormatException)

{.Show("Invalid data!\nPlease, enter other data.", "Error converting");;

}N = listBox1.Items.Count;[,] mat = new decimal[2, N];

//перводим listBox в массив чисел(int i = 0; i < N; i++)

{x, y;s = listBox1.Items[i].ToString();(s, out x, out y);[0, i] = x;[1, i] = y;

}

//сортируем массив по Х

//за одно проверяем на наличие одинаковых точек

for (int i=0; i<mat.GetLength(1)-1; i++)(int j = i+1; j < mat.GetLength(1); j++)

{(mat[0, i] == mat[0, j])

{.Text = String.Format(

"Error points: \nx={0:F5}; y={1:F5}\nx={2:f5}; y={3:f5}",[0,i],mat[1,i],mat[0,j],mat[1,j]);;

}buf1,buf2;(mat[0, i] > mat[0, j])

{= mat[0, i]; buf2 = mat[1, i];[0, i] = mat[0, j]; mat[1, i] = mat[1, j];[0, j] = buf1; mat[1, j] = buf2;

}

}

//находим проихводную, если это возможно(dif.isSolve(mat, x1))

{

//timer для подчета тактов процессораtimer = new Stopwatch();

timer.Start();.Text = String.Format("f'(x) = {0:F5}", dif.Solve(mat, x1));.ShowAnswer();.Stop();.Text += String.Format("\nTime: {0} ticks.", timer.ElapsedTicks);.Invalidate();

}label1.Text = "Нет решения";

}void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

//переключаем интерфейсную ссылку на нужный режим

if (comboBox1.SelectedIndex == 0) dif = Razn;dif= Interp;

}void panel1\_Paint(object sender, PaintEventArgs e)

{buffer;context;= BufferedGraphicsManager.Current;.MaximumBuffer = new System.Drawing.Size(panel1.Width + 1, panel1.Height + 1);rec = new Rectangle(0, 0, panel1.Width, panel1.Height);= context.Allocate(e.Graphics, rec);

//ТУТ ДОЛЖНА БЫТЬ ОТРИСОВКА КООДРИНАТНОЙ СЕТКИ

//И графика из точек

//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++zoom = form2.trackBar1.Value;font = new System.Drawing.Font("Arial", 6);

//используемые кистиpen = new Pen(Color.Black); //для секти

Pen penRed = new Pen(Color.Red);//для графика по точкам

//используемые заливки

Brush brush = new SolidBrush(Color.Black); //для сетки

//Отрисовка системы коодринат.Graphics.Clear(Color.White);Xs = (int)(form2.x\*zoom);Ys = (int)(form2.y \* zoom);maxX = panel1.Width / 2 ;maxY = panel1.Height / 2 ;.Graphics.TranslateTransform(maxX-Xs, maxY+Ys);

//рисуем оси Х и Y, а так же разметку

buffer.Graphics.DrawLine(pen, 0, maxY-Ys, 0, -maxY-Ys);.Graphics.DrawLine(pen, maxX+Xs, 0, -maxX+Xs, 0);

//int stepX =StepM = panel1.Width / zoom;StepMin = (int)(maxX - Xs)/zoom;(int j = 0; j < StepM; j++)

{i = j-StepMin;(i == 0) continue;.Graphics.DrawLine(pen, i \* zoom, 3f, i \* zoom, -3f);.Graphics.DrawString(String.Format("{0}", i), font, brush, i \* zoom, -12f);

}= panel1.Height / zoom;= (int)(maxY + Ys) / zoom;(int j = 0; j < StepM; j++)

{i = j - StepMin;.Graphics.DrawLine(pen, 3f, i \* zoom, -3f, i \* zoom);.Graphics.DrawString(String.Format("{0}", -i), font, brush, -12f, i \* zoom);

}[] p1 = new PointF[3];[0] = new PointF(0f, -(float)(maxY + Ys));[1] = new PointF(4f, -(float)(maxY + Ys) + 10f);[2] = new PointF(-4f, -(float)(maxY + Ys)+ 10f);.Graphics.DrawString("Y", font, brush, 12f, -(maxY + Ys));.Graphics.FillPolygon(brush, p1);[0] = new PointF((float)maxX+Xs, 0f);[1] = new PointF((float)(maxX+Xs)- 10f, 4f);[2] = new PointF((float)(maxX +Xs) - 10f, -4f);.Graphics.DrawString("X", font, brush, (maxX + Xs) - 12f, 12f);.Graphics.FillPolygon(brush, p1);

//конец отрисовки коодринат

//Начинаем сторить график.

int N = listBox1.Items.Count;[] points = new PointF[N];(int i = 0; i < N; i++)

{x, y;s = listBox1.Items[i].ToString();(s, out x, out y);[i].X = (float)x\*zoom;[i].Y = -(float)y\*zoom;

}(int i=0; i<N-1; i++)(int j = i+1; j < N; j++)

{(points[i].X == points[j].X)

{.Text = String.Format(

"Error points: \nx={0:F5}; y={1:F5}\nx={2:f5}; y={3:f5}",[i].X, points[i].Y, points[j].X, points[j].Y);;

}buf;(points[i].X > points[j].X)

{= points[i];[i] = points[j];[j] = buf;

}

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

{.Graphics.DrawLines(penRed, points);

}

//построили график

//сторим полином, если активирован нужный режим.Paint(buffer, zoom);

//построили, если что))

//Выводим на экран, что получилось).Render(e.Graphics);

//и очищаем память

buffer.Dispose();

}void graficToolStripMenuItem\_Click\_1(object sender, EventArgs e)

{.Show();

}void Form1\_ClientSizeChanged(object sender, EventArgs e)

{.Invalidate();

}void textBox3\_KeyDown(object sender, KeyEventArgs e)

{(e.KeyValue == 13) this.button3\_Click(this, e);

}

}

}

Модуль From2.cs:System;System.Collections.Generic;System.ComponentModel;System.Data;System.Drawing;System.Linq;System.Text;System.Windows.Forms;Kursavik\_sem\_3

{partial class Form2 : Form

{float x;float y;mainForm;Form2()

{();

}Form2(Form1 ptr)

{= ptr;();.Text = "Grafic";= 0;= 0;

}void trackBar1\_Scroll(object sender, EventArgs e)

{.panel1.Invalidate();

}void Form2\_Load(object sender, EventArgs e)

{.Text = "Grafic";

}void textBox1\_TextChanged(object sender, EventArgs e)

{

{= Convert.ToSingle(textBox1.Text);= Convert.ToSingle(textBox2.Text);.panel1.Invalidate();

}(FormatException)

{.Text = Convert.ToString(x);.Text = Convert.ToString(y);.Show("Invalid data!\nPlease, enter other data.", "Error converting");

}

}void textBox1\_KeyDown(object sender, KeyEventArgs e)

{(e.KeyValue == 13) this.textBox1\_TextChanged(this, e);

}void Form2\_FormClosing(object sender, FormClosingEventArgs e)

{.Cancel = true;.Hide();

}

}

}

Модуль Differ.cs:System;System.Collections.Generic;System.Linq;System.Text;System.Drawing;System.Windows.Forms;Kursavik\_sem\_3

{

//интерфейс для работы с разными методамиIDiffer

{

//метод проверяет возможность решения

bool isSolve(decimal[,] matrix, decimal point);

//метод находит производнуюSolve(decimal[,] matrix, decimal point);

//метод рисует на графике что-то если это нужно

void Paint(BufferedGraphics buffer, int zoom);

//показывает решениеShowAnswer();

}

//метод разностных производных

class RaznDiffer :IDiffer

{yL, yR, y, h;

bool R, L, C; //используемый метод(для ShowAnsewr)

Label lab; //куда выводитьRaznDiffer(Label l)

{= l;

}bool isSolve(decimal[,] matrix, decimal point)

{(matrix.GetLength(0) != 2) return false;PointFind = matrix[0, 0] == point; ;StepRight = true;step = matrix[0, 1] - matrix[0, 0];(int i = 1; i < matrix.GetLength(1); i++)

{

//проверям равномерность сетки

//а также наличие данной точки в массиве

decimal s = matrix[0, i] - matrix[0, i - 1];(Math.Abs(step-s)>0.0000001M)

{= false;;

}(matrix[0, i] == point) PointFind = true;

}PointFind&&StepRight;

}decimal Solve(decimal[,] matrix, decimal point)

{pointInd=0;N=matrix.GetLength(1);

decimal h = matrix[0, 1] - matrix[0, 0];

//ищем номер точки для дифференцирования

for (int i=0; i<N; i++)(point == matrix[0, i])

{= i;

break;

}

//используем формулы, в зависимости

//от найденой точки

if (pointInd == 0)

{= true;= C = false;= matrix[1, 0];= matrix[1, 1];.h = h;(matrix[1, 1] - matrix[1, 0])/h;

}(pointInd == N - 1)

{= true;= C = false;= matrix[1, pointInd];= matrix[1, pointInd - 1];.h = h;(matrix[1, pointInd] - matrix[1, pointInd-1]) / h;

}(pointInd != 0 && pointInd != N - 1)

{= true;= L = false;= matrix[1, pointInd + 1];= matrix[1, pointInd - 1];.h = h;(matrix[1, pointInd + 1] - matrix[1, pointInd - 1]) / (2 \* h);

}0;

}void Paint(BufferedGraphics buffer, int zoom)

{

//тут не нужно ничего рисовать

}

public void ShowAnswer()

{.ResetText();

//выводим текст в зависимости от точки

if (R) lab.Text = String.Format("f'(x) = ({0:f5}-{1:f5})/{2:F5}", yR, y, h);(L) lab.Text = String.Format("f'(x) = ({0:f5}-{1:f5})/{2:F5}", y, yL, h);(C) lab.Text = String.Format("f'(x) = ({0:f5}-{1:f5})/{2:F5}", yR, yL, 2\*h);

}

}

//с использованием интерполяцииInterDiffer : IDiffer

{

//набор точек и значений

//используемых в предыдущй раз

decimal[,] interMat;

//получившаяся фукнция[] intrFunc;

//производная функиции[] difPolinom;lab;CompareMat(decimal[,] matrix)

{

//метод сравнивает 2-е матрицы(matrix.GetLength(0) != interMat.GetLength(0)) return false;(matrix.GetLength(1) != interMat.GetLength(1)) return false;(int i = 0; i < matrix.GetLength(0); i++)(int j = 0; j < matrix.GetLength(1); j++)(matrix[i, j] != interMat[i, j]) return false;true;

}CreateMatInterol(decimal[,] matrix)

{

//метод сторит матрицу,

//необходимую для получения полинома

int N = matrix.GetLength(0);(int i = 0; i < N; i++)

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

{(j == 0)

{[i, j] = 1;;

}[i, j] = matrix[i, j - 1] \* interMat[0, i];

}

}(int i = 0; i < N; i++) matrix[i, N] = interMat[1, i];

}InterDiffer() { }InterDiffer( Label lab)

{.lab = lab;

}bool isSolve(decimal[,] matrix, decimal point)

{(matrix.GetLength(0) != 2) return false;true;

}decimal Solve(decimal[,] matrix, decimal point)

{

//сравниваем матрицы, если они одинаковые

//то расчитывать полином заново не нужно

if (interMat==null ||!CompareMat(matrix) )

{= matrix;N = matrix.GetLength(1);[,] iMat = new decimal[N,N+1];[] ans = new decimal[N];[] dif = new decimal[N-1];

//создаем матрицу для решения(iMat);

//и решаем ее.GetSolve(iMat, ans);

//а потом берем производную от решения

MathBase.GetDifferPolinom(ans, dif);= ans;= dif;

}MathBase.Fx(difPolinom,point);

}void Paint(BufferedGraphics buffer, int zoom)

{penRed = new Pen(Color.SeaGreen);minX = (float)interMat[0, 0];maxX = (float)interMat[0, interMat.GetLength(1) - 1];(float x = -1f; x < maxX; x += 0.01f).Graphics.DrawLine(penRed, x \* zoom,

(float)MathBase.Fx(intrFunc, (decimal)x) \* zoom,

(x + 0.01f) \* zoom,

(float)MathBase.Fx(intrFunc,(decimal)(x + 0.01f)) \*zoom);

}void ShowAnswer()

{.ResetText();.Text = "Interpolation Polynomial:\n";(int i = intrFunc.Length - 1; i > 0; i--)

{.Text += String.Format("{0:F5}x^{1} + ", intrFunc[i], i);

}.Text += String.Format("{0:F5}\n", intrFunc[0]);.Text += "Derivative of Polynomial:\n";(int i = difPolinom.Length - 1; i > 0; i--)

{.Text += String.Format("{0:F5}x^{1} + ", difPolinom[i], i);

}.Text += String.Format("{0:F5}\n", difPolinom[0]);

}

}

}

Модуль MathBase.cs:System;System.Collections.Generic;System.Linq;System.Text;Kursavik\_sem\_3

{MathBase

{

//метод копирует матрицу MatIn в матрицу MatOutvoid copy(decimal[,] MatIn, decimal[,] MatOut)

{a = MatIn.GetLength(0);b = MatIn.GetLength(1);(int i = 0; i < a; i++)(int j = 0; j < b; j++) MatOut[i, j] = MatIn[i, j];

}

//метод обменивает указанные строки матрицыvoid swap(decimal[,] matrix, int a, int b)

{len = matrix.GetLength(1);(a != b)(int i = 0; i < len; i++)

{buf;= matrix[a, i];[a, i] = matrix[b, i];

matrix[b, i] = buf;

}

}

//метод делает матрицу диаганальной

static void GetDiaganal(decimal[,] matrix)

{a = matrix.GetLength(0);b = matrix.GetLength(1);(int i = 0; i < a - 1; i++)

{max = i, sw = i;maxVol = Math.Abs(matrix[i, i]);

//находим максимальный элемент в столбце(int j = i; j < a; j++)

{(Math.Abs(matrix[j, i]) > maxVol)

{= Math.Abs(matrix[j, i]);= j;

}

}

//меняем строки местами(matrix,max,sw);kof;

//вычитаем строки.(int j = i + 1; j < a; j++)

{= matrix[j, i] / matrix[i, i];(int k = i; k < b; k++)

{[j, k] -= kof \* matrix[i, k];

}

}

}

}

//метод решает СЛАУstatic void GetSolve(decimal[,] inMat, decimal[] Solve)

{[,] buffer = new decimal[inMat.GetLength(0),inMat.GetLength(1)];(inMat, buffer);(buffer);a = inMat.GetLength(0);b = inMat.GetLength(1);(int i = a - 1; i >= 0; i--)

{resul = buffer[i, b - 1];(int j = b - 2; j > i; j--) resul -= buffer[i, j] \* Solve[j];[i] = resul / buffer[i, i];

}

}

//метод берет производную от полиномаstatic void GetDifferPolinom(decimal[] inMat, decimal[] outMat)

{(int i = 0; i < inMat.Length - 1; i++)[i] = inMat[i + 1] \* (i + 1);

}

//метод считает значение полиномаstatic decimal Fx(decimal[] polinom, decimal x)

{ans = 0;(int i = 0; i < polinom.Length; i++)

{+=polinom[i] \* (decimal)(Math.Pow((double)x,(double)i));

}ans;

}

}

}

# Приложение Б

Руководствопользователя

Описание программы:

Данная программа предназначена для нахождения производной функции, заданной таблично.

Программа может рассчитывать производную двумя способами: с помощью разностных формул и с помощью интерполяционного многочлена Лагранжа.

Системные требования:

Процессор: 800 МГц и выше.

Операционная система: Windows 7 и выше.

Дополнительное ПО:.NET Framework 4.

Стартовое окно.

Во время запуска программы вы увидите следующее окно, показанное на рисунке Б.1:



Рисунок Б.1 - Стартовое окно программы

Данное окно содержит все необходимое для нахождения значения производной в нужной точке.

В данном окне можно:

) Задать множество точек функции.

) Задать точку, в которой нужно найти производную.

) Задать метод решения задачи.

) Ознакомиться с результатами решения в текстовом и графическом видах.

Добавление точки.

Для того, чтобы добавить точку в множеству точек функции необходимо в поля "X" и "Y" (на рисунке Б.2 обозначены 1) ввести необходимые данные. После этого нажать кнопку "Add" (на рисунке обозначена 2) или клавишу Enter. Если в поле будут введены не корректные данные, будет показано соответствующее предупреждение, а данные не будут занесены в таблицу.



Рисунок Б.2 - Добавление точки функции

Удаление точки

Если вы убрать какие-то точки из расчета, то необходимо выбрать соответствующие точки в списке "Point List" (на рисунке обозначен 1). После этого необходимо нажать клавишу "Delete" (на рисунке Б.3 обозначена 2). После этого указанные точки будут удалены. Если не выбрана ни одна точка, то при попытке нажатия клавишу "Delete" будет выведено предупреждение.



Рисунок Б.3 - Удаление точки из списка

Выбор метода

Для того, чтобы выбрать интересующий вас метод решения, необходимо на нажать на выпадающий список с заголовком "Selected method" (выделен на рисунке Б.4) и выбрать в списке один из двух методов.derivation - использует разностные формулы вычисления производной.Interpolation - использует интерполяционный многочлен Лагранжа.



Рисунок Б.4 - Выбор метода дифференцирования

Выбор точки для дифференцирования

Для того, чтобы произвести расчет производной, нужно указать точку, в которой будет производиться расчет.

В поле ввода с заголовком "Point for differentiation" (выделено на рисунке Б.5) нужно ввести точку. Если будет введено некорректное значение, при попытке рассчитать производную, будет показано предупреждающее сообщение, а расчет не будет произведен. Однако, если точка не будет удовлетворять условиям метода, расчет будет произведен, но вместо результатов будет выведено: "Нет решения".

Для того, чтобы начать расчет, нужно нажать кнопку "Find derivative" или клавишу Enter.



Рисунок Б.5 - Выбор точки дифференцирования

Результаты работы

После завершения расчетов по поле, выделенном на рисунке Б.6 будут отображаться результаты, а также некоторая информация о решении задачи.

Также в данном поле может быть информация о невозможности решить задачу данным методом, или информация об ошибке, когда одному аргументу функции соответствует два значения функции.



Рисунок Б.6 - Результаты расчета

Окно настройки графика

Для того, чтобы открыть окно(рисунок Б.8) управления графиком, в меню необходимо перейти Setting->Grafic (рисуонк Б.7).



Рисунок Б.7 - Включение окна настройки графика

В окне настройки графика возможно:

) Задать смещение графика по осям.

) Задать масштаб графика.



Рисунок Б.8 - Настройка графика

Настройки графика.

Для настройки смещения графика по осям, необходим в поля (выделены 1 на рисунке Б.9) прописать необходимое смещения.

Для масштабирования графика необходимо двигать ползунок (выделен 2 на рисунке Б.9).



Рисунок Б.9 - Выбор параметров графика