МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования



НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра «Вычислительные системы и технологии»

# ОТЧЁТ

по лабораторной работе №3

" ИНТЕРПОЛИРОВАНИЕ ФУНКЦИИ

МНОГОЧЛЕНОМ НЬЮТОНА И МНОГОЧЛЕНОМ ЛАГРАНЖА"

по дисциплине

*Вычислительная математика*

(наименование дисциплины)

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_Панкратова А.З.\_\_\_

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_Халеев А.А. \_

(подпись) (фамилия, и.,о.)

\_\_\_\_\_\_\_21-ВМз-4\_\_\_\_\_\_

(шифр группы)

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород

2023

**Тема работы:**

Интерполирование функции многочленом Ньютона и многочленом Лагранжа

**Цель работы**:

Изучить численные методы и алгоритмы решения задачи интерполяции функций

**Постановка задачи:**

Реализовать изученные алгоритмы интерполяции и провести сравнение методов

**Вариант №7:**

1. Найти приближенное значение функции при данных значениях аргумента с помощью интерполяционного многочлена Лагранжа, если функция задана в неравноотстоящих узлах таблицы, оценить погрешность

|  |  |  |  |
| --- | --- | --- | --- |
| x | y |  |  |
| 0.43 | 1.63597 | 0.702 | 0.503 |
| 0.48 | 1.73234 | 0.512 | 0.441 |
| 0.55 | 1.87686 | 0.645 | 0.602 |
| 0.62 | 2.03345 | 0.736 | 0.732 |
| 0.70 | 2.22846 |  |  |
| 0.75 | 2.35973 |  |  |

1. Используя первую или вторую интерполяционную формулу Ньютона, вычислить значения функции при данных значениях аргумента 2yi

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | y | Значение аргумента | | | |
| 1.340 | 4.25562 | x1 | x2 | x3 | x4 |
| 1.345 | 4.35325 | 1.3617 | 1.3921 | 1.3359 | 1.400 |
| 1.350 | 4.45522 |  |  |  |  |
| 1.355 | 4.56184 |  |  |  |  |
| 1.360 | 4.67344 |
| 1.365 | 4.79038 |
| 1.370 | 4.91306 |
| 1.375 | 5.04192 |
| 1.380 | 5.17744 |
| 1.385 | 5.32016 |
| 1.390 | 5.47069 |
| 1.395 | 5.62968 |

***Теория:***

В инженерных расчетах часто требуется установить функцию f(x) для всех значений х отрезка [a,b] , если известны ее значения в некотором конечном числе точек этого отрезка. Одним из способов приближения функции является интерполяция. Задача интерполяции может возникнуть в практике при:

1) интерполировании табличных данных;

2) получении функциональной зависимости по экспериментальным данным, представленным в табличной форме;

3) замене сложной с вычислительной точки зрения функции, более простой зависимостью;

4) при дифференцировании и интегрировании

Пусть на отрезке [x0 ,xn ] заданы n+1 точки x0 , x1 , x2 ,...,xn , называемые узлами интерполяции, и значения некоторой интерполируемой функции y=f(x) в этих точках, т.е. имеется таблица экспериментальных значений функции y=f(x); y0 , y1 , y2 .....yn, где y0=f(x0 ); y1=f(x1 ); ...; yn=f(xn )

Постановка задачи: Требуется найти значения этой функции для промежуточных значений аргумента, не совпадающих с приведенными в таблице. Получить аналитическое выражение функции y= f(x) по таблице ее значений часто бывает невозможно.

Поэтому вместо нее строят другую функцию, которая легко вычисляется и имеет ту же таблицу значений, что и f(x), т.е. Pm(x0 )=f(x0)=y0, …Pm (xi)=f(xi )=yi , где i = 0,1,2, ... , n. Такую задачу называют задачей интерполирования. Точки xi называются узлами интерполяции; функция f(x) называется интерполируемой функцией; многочлен Pm (x) называется интерполяционным многочленом.

***Интерполяционный многочлен Лагранжа для не равностоящих узлов***

*Пусть функция f задана таблицей:*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *x* | *xо* | *x\* | *...* | xn |
| *f(х)* | *y0* | *y1* | ... | yn |

Построим интерполяционный многочлен Ln(х), степень которого не больше n и для которого выполнены условия

*F(хо)=у0, F(х1)=у1,* .... *F(хn)=yп.* , (1)

Будем искать *Ln(х)* в виде

*Ln(x)=l0(x)+l1(x)+l2(x)+………+ln(x)* , (2)

где *li(х) —* многочлен степени *п,*

Очевидно, что требование (3) с учетом (2) обеспечивает выполнение условий (1).

Многочлены *li(х)* составим следующим способом:

*li(x)=ci(x-x0)(x-x1)·… ·(x-xi-1)(x-xi+1) ·…·(x-xn),* (4)

где сi - постоянный коэффициент, значение которого можно найти из первой части условия (3). Окончательно получим:

) (5)

Это и есть интерполяционный многочлен Лагранжа для не равностоящих узлов.

***Исходные данные:***

*Функция задана таблицей:*

|  |  |
| --- | --- |
| x | y |
| 0.43 | 1.63597 |
| 0.48 | 1.73234 |
| 0.55 | 1.87686 |
| 0.62 | 2.03345 |
| 0.70 | 2.22846 |
| 0.75 | 2.35973 |

*Искомое значение аргумента:*

|  |  |
| --- | --- |
|  |  |
| 0.702 | 0.503 |
| 0.512 | 0.441 |
| 0.645 | 0.602 |
| 0.736 | 0.732 |

***Решение:***

Из таблицы следует, что n=5, (т.е. степень многочлена будет не выше чем пятая);

Здесь x0=0.43, x1=0.48, x2=0.55, x3=0.62, x4=0.70, x5=0.75

Используя формулу интерполяционного многочлена Лагранжа получаем:

Таким образом, приближающая функция для функции *f* заданной таблицей имеет следующий вид:

***Решение в Excel:***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | y | x1 | y1 | x2 | y2 |
| 0,43 | 1,6360 | 0,702 | 2,41948 | 0,503 | 1,87399 |
| 0,48 | 1,7323 | 0,512 | 1,89585 | 0,441 | 1,73006 |
| 0,55 | 1,8769 | 0,645 | 2,24943 | 0,602 | 2,12869 |
| 0,62 | 2,0335 | 0,736 | 2,52657 | 0,732 | 2,51375 |
| 0,70 | 2,2285 |  |  |  |  |
| 0,75 | 2,3597 |  |  |  |  |

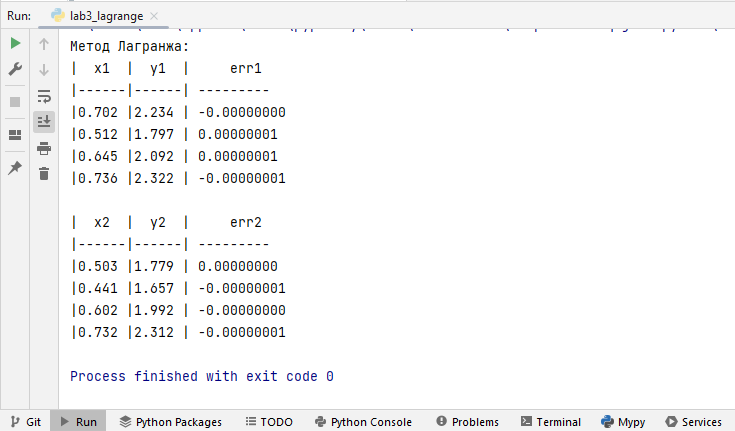
***Программа на Python:***

import operator  
import warnings  
from functools import reduce  
from math import factorial  
  
from scipy.misc import derivative  
  
warnings.filterwarnings("ignore", category=DeprecationWarning)  
  
  
def lagrange\_interpolation(x\_values, y\_values, x):  
 *"""  
 Расчитывает значение в точке x для интерполирующего многочлена  
 для данной последовательности точек, используя метод Лагранжа.  
  
 Аргументы:  
 x\_values (list): список x-координат точек данных.  
 y\_values (list): список y-координат точек данных.  
 x (float): точка, в которой вычисляется полином.  
  
 Возвращает:  
 float: значение интерполирующего полинома в точке x.  
 """* n = len(x\_values)  
 p = 0 *# Инициализация интерполирующего полинома* for i in range(n):  
 L = 1 *# Инициализация полинома Лагранжа* for j in range(n):  
 if i != j:  
 L \*= (x - x\_values[j]) / (x\_values[i] - x\_values[j])  
 p += y\_values[i] \* L  
  
 return p  
  
  
def lagrange\_error(x\_values, y\_values, x):  
 *"""  
 Вычисляет погрешность в интерполяции Лагранжа.  
  
 Аргументы:  
 x\_values (list): список x-координат точек данных.*

*y\_values (list): список y-координат точек данных.  
 x (float): точка, в которой вычисляется ошибка.  
  
 Возвращает:  
 float: оценка погрешности интерполяции Лагранжа в точке x.  
 """* n = len(x\_values)  
 product = reduce(operator.mul, [(x - xi) for xi in x\_values])  
   
 func = lambda x: lagrange\_interpolation(x\_values, y\_values, x)  
 *# значение (n+1)-ой производной в точке x.* nth\_derivative = derivative(func, x0=x, n=n - 1, order=n + [1, 0][n % 2])  
  
 return nth\_derivative \* product / factorial(n + 1)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 *# Вычисление значений в точках* x = [0.43, 0.48, 0.55, 0.62, 0.70, 0.75]  
 y = [1.63597, 1.73234, 1.87686, 2.03345, 2.22846, 2.35973]  
  
 x1 = [0.702, 0.512, 0.645, 0.736]  
 y1 = [lagrange\_interpolation(x, y, x1[i]) for i in range(len(x1))]  
 err1 = [lagrange\_error(x, y, x1[i]) for i in range(len(x1))]  
  
 x2 = [0.503, 0.441, 0.602, 0.732]  
 y2 = [lagrange\_interpolation(x, y, x2[i]) for i in range(len(x2))]  
 err2 = [lagrange\_error(x, y, x2[i]) for i in range(len(x2))]

print("Метод Лагранжа:")  
 print("| x1 | y1 | err1")  
 print("|------|------| ---------")  
 [print(  
 f"|{x1[i]} |{y1[i]:.3f} | {err1[i]:.8f} ")  
 for i in range(len(x1))]  
 print("\n| x2 | y2 | err2")  
 print("|------|------| ---------")  
 [print(  
 f"|{x2[i]} |{y2[i]:.3f} | {err2[i]:.8f} ")  
 for i in range(len(x1))]

Вывод программы:



***Вторая интерполяционная формула Ньютона***

Для получения второй интерполяционной формулы Ньютона нужно представить искомый многочлен в виде:

(9)

Полагая значения x в формуле:

) (5)

равными поочередно: x=xn,…x=x0, можно вычислить все коэффициенты этого многочлена:

Если подставить найденные коэффициенты в формулу (9), получится вторая интерполяционная формула Ньютона. Введем новую переменную t:

(10)

Это и есть обычный вид второй интерполяционной формулы Ньютона. Вторая интерполяционная формула Ньютона применяется при построении многочленов для узлов, расположенных ближе к правому концу таблицы (интерполирование назад) и для узлов, расположенных вне таблицы за правым концом (экстраполирование вперед).

***Исходные данные:***

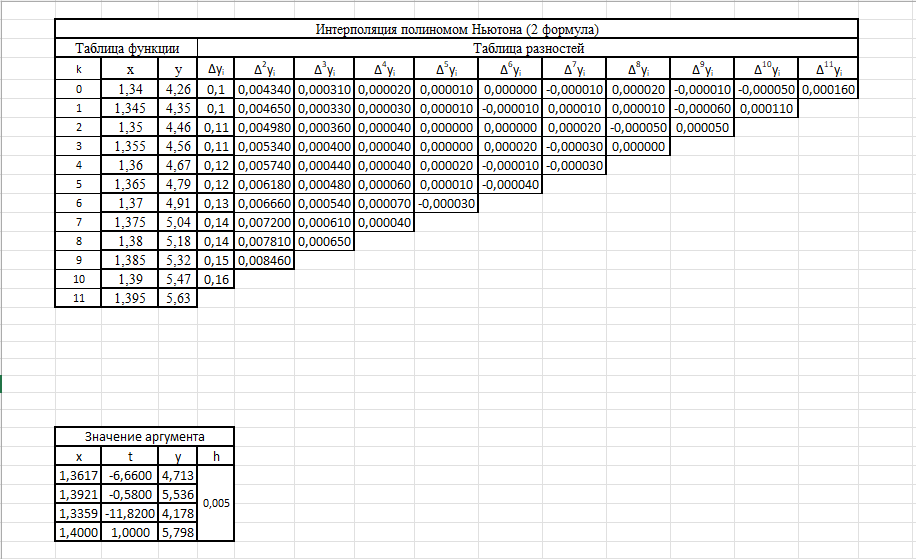
*Функция задана таблицей:*

|  |  |
| --- | --- |
| x | y |
| 1.340 | 4.25562 |
| 1.345 | 4.35325 |
| 1.350 | 4.45522 |
| 1.355 | 4.56184 |
| 1.360 | 4.67344 |
| 1.365 | 4.79038 |
| 1.370 | 4.91306 |
| 1.375 | 5.04192 |
| 1.380 | 5.17744 |
| 1.385 | 5.32016 |
| 1.390 | 5.47069 |
| 1.395 | 5.62968 |

*Искомое значение аргумента:*

|  |  |  |  |
| --- | --- | --- | --- |
| x1 | x2 | x3 | x4 |
| 1.3617 | 1.3921 | 1.3359 | 1.400 |

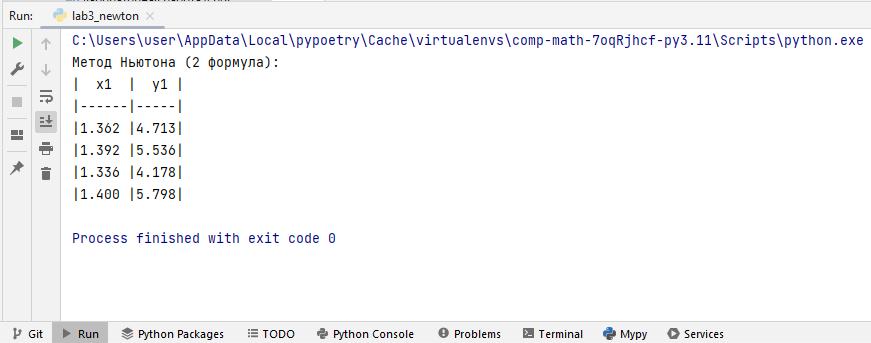
***Решение в Excel:***



***Программа на Python:***

import operator  
import warnings  
from functools import reduce  
from math import factorial  
  
from scipy.misc import derivative  
  
warnings.filterwarnings("ignore", category=DeprecationWarning)  
  
  
def newton\_interpolation(x, y, x0):  
 *"""  
 Вычисляет значение интерполированной функции в точке x0 с использованием второй интерполяционной формулы Ньютона.  
  
 Параметры:  
 x (list): Список точек интерполяции.  
 y (list): Список значений функции в точках интерполяции.  
 x0 (float): Точка, в которой требуется вычислить значение интерполированной функции.  
  
 Возвращает:  
 y0 (float): Значение интерполированной функции в точке x0.  
 """* n = len(x)  
  
 *# Инициализация разделенной разности* fdd = [[0 for x in range(n)]  
 for y in range(n)]  
 y = y.copy()  
  
 *# Заполнение нулевой строки таблицы разделенных разностей* for i in range(n):  
 fdd[i][0] = y[i]  
  
 *# Заполнение других строк таблицы разделенных разностей* for i in range(1, n):  
 for j in range(n - i):  
 *# Вычисление i-ой разделенной разности* fdd[j][i] = (fdd[j + 1][i - 1] - fdd[j][i - 1]) / (x[i + j] - x[j])  
  
 *# Вычисление значения интерполированной функции* y0 = fdd[0][0]  
 for i in range(1, n):  
 prod = 1  
 for j in range(i):  
 prod \*= (x0 - x[j])  
 *# Добавление i-ого слагаемого в сумму* y0 += prod \* fdd[0][i]  
  
 return y0  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 *# Вычисление значений в точках* x = [1.34, 1.345, 1.35, 1.355, 1.36, 1.365, 1.37, 1.375, 1.38, 1.385, 1.39, 1.395]  
 y = [4.25562, 4.35325, 4.45522, 4.56184, 4.67344, 4.79038, 4.91306, 5.04192, 5.17744, 5.32016, 5.47069, 5.62968]  
  
 x1 = [1.3617, 1.3921, 1.3359, 1.4]  
 y1 = [newton\_interpolation(x, y, x1[i]) for i in range(len(x1))]  
  
 print("Метод Ньютона (2 формула):")  
 print("| x1 | y1 |")  
 print("|------|-----|")  
 [print(  
 f"|{x1[i]:.3f} |{y1[i]:.3f}|")  
 for i in range(len(x1))]

Вывод программы:



**Вывод:**

В данной лабораторной работе были изучены следующие методы численного решения задач интерполяции функций:

- интерполяция полиномом Лагранжа

- интерполяция полиномами Ньютона

Также были реализованы программные алгоритмы интерполяции.

При интерполировании с помощью многочлена Лагранжа, для функции заданной в не равноотстоящих узлах таблицы, была вычислена погрешность, в среднем не превышающая:

1e-8 = 0.00000001

Таким образом, метод интерполяции полиномом Лагранжа является точным до 8 знака после запятой.

Значения, полученные при ручном счете и программно являются числами одного порядка и совпадают до 3го знака.