|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** ***ИУК «Информатика и управление»***

**КАФЕДРА** \_\_***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

**ЛАБОРАТОРНАЯ РАБОТА №1**

**«Методы численного интегрирования»**

**ДИСЦИПЛИНА: «Моделирование»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-62Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Карельский М.К. )  (Подпись) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Никитенко У.В. )  (Подпись) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |

Калуга, 2023

**Цель:** сформировать практические навыки анализа возможностей построения и выделения наиболее важных свойств объектов моделей для моделирования и использования специализированных программных пакетов и библиотек для стандартных вычислений и визуализации результатов численной аппроксимации производных и обоснования выбора алгоритма аппроксимации.

**Задачи:** изучить методы аппроксимации производных, методы оценки точности аппроксимации, методы повышения точности аппроксимации, количественные характеристики методов, написать программы, указанные в вариантах.

**Вариант 16**

1. Вычислить приближенно значения

а) первой производной функции y = f(x) с порядком погрешности O(h) (обозначим ) и O(h2) (обозначим ) при i = 0, 1, . . . , n.

б) второй производной функции y = f(x) с порядком погрешности O(h) (обозначим ) при i = 1, . . . , n − 1.

Напечатать таблицу значений узлов, «точных» значений производных в узлах, приближенных значений производных и их разностей (фактические погрешности). Проверить результаты на многочленах соответствующих степеней. Объяснить полученные результаты.

1. Пользуясь формулой

в точке x = 1 вычислить разностную производную второго порядка аппроксимации функции e4x, последовательно уменьшая шаг h (например, вдвое) до тех пор, пока фактическая погрешность не начнет возрастать. Определить h оптимальное экспериментально и теоретически, объяснить полученные результаты.

Значения функции округлять до 5-ого знака после запятой, т. е. ε = 5·10−6

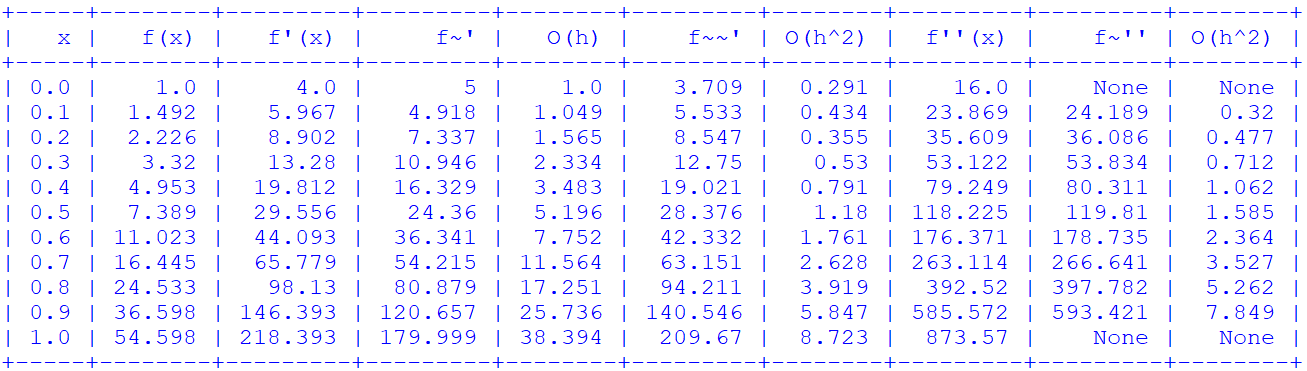
1. Предполагается заданной таблица значений функции в равноотстоящих узлах xi, i = 0, ..., n.

Требуется дифференцированием интерполяционного многочлена в форме Ньютона получить формулу численного дифференцирования для вычисления приближенного значения второй производной с первым порядком аппроксимации в точке x = xn. Получить выражение для погрешности. Применить формулу для вычисления производной, сравнить с точным значением.

**Решение:**

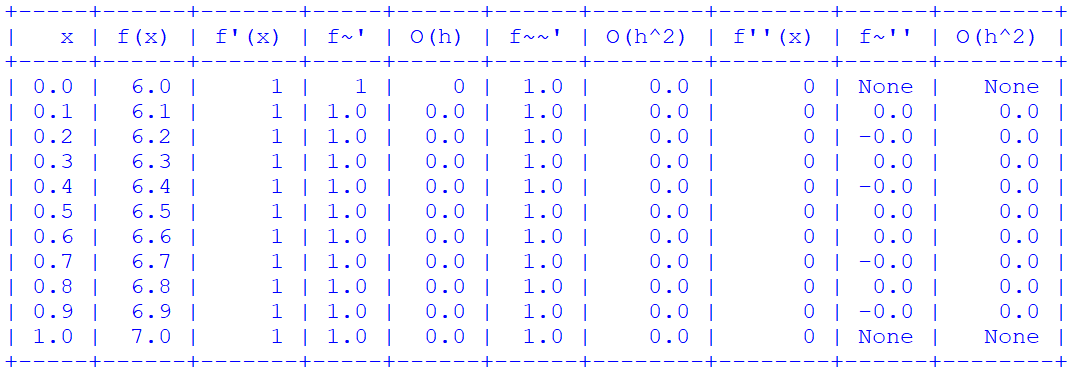
***Задача 1:***

Результаты для исходной функции:



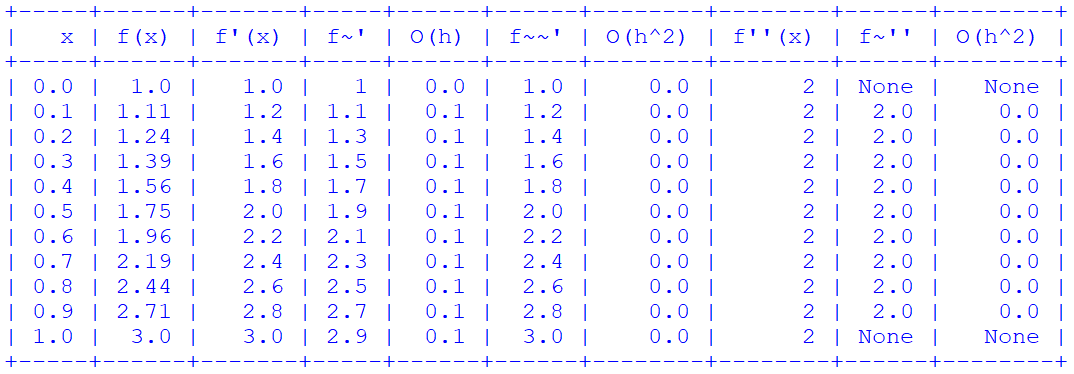
**Рис. 1.** f(x) = e4x

Для многочлена первой степени приближенные значения совпадают с аналитическими:



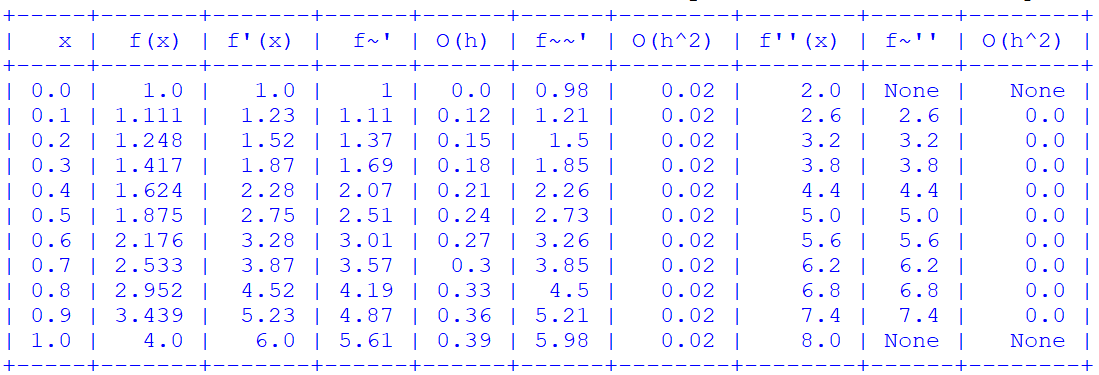
**Рис. 2.** f(x) = x + 6

Приближенные значения первой производной первого порядка погрешности многочлена второй степени имеют погрешность.



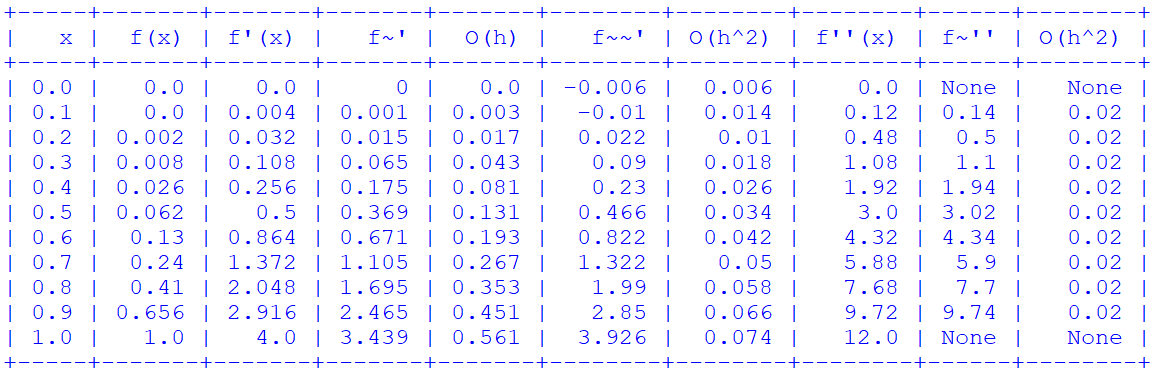
**Рис. 3.** f(x) = x2 + x + 1

Погрешность также появляется в приближенных значениях первой производной второго порядка погрешности многочлена третьей степени:



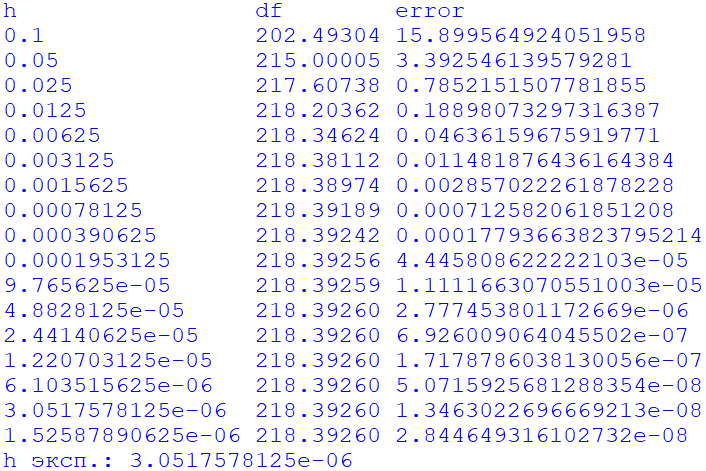
**Рис. 4.** f(x) = x3 + x2 + x + 1

Наконец, погрешность появляется в приближенных значениях второй производной второго порядка погрешности многочлена четвертой степени:



**Рис. 5.** f(x) = x4

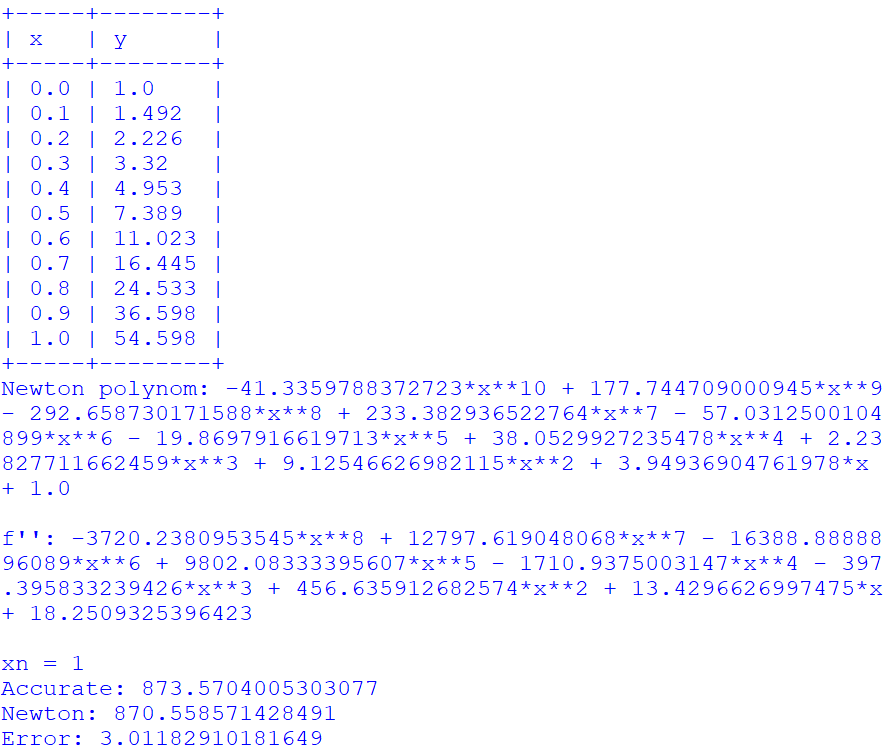
***Задача 2:***



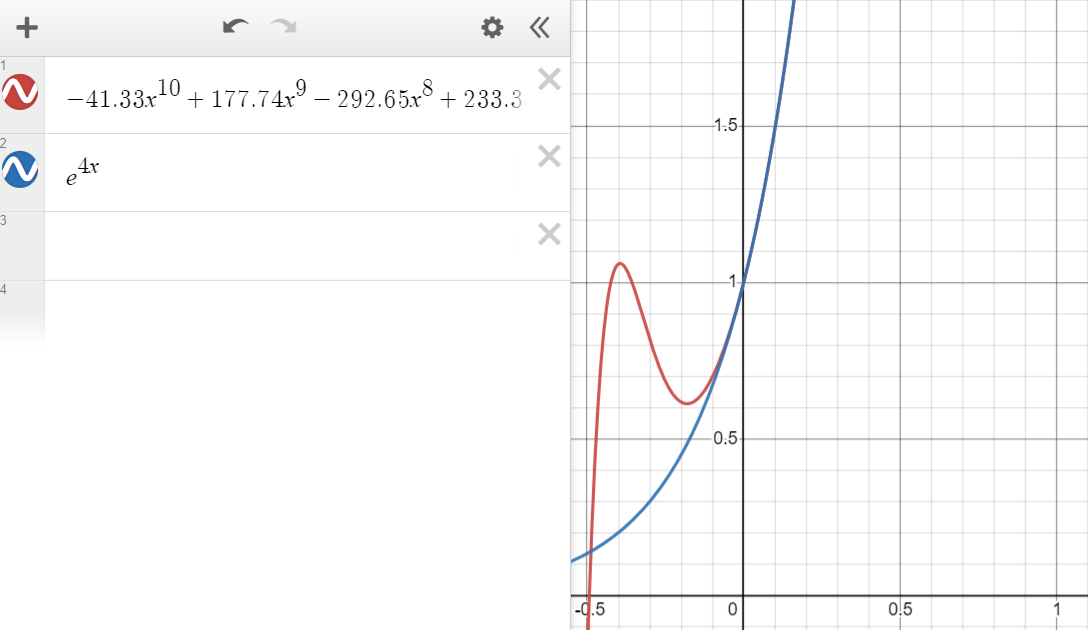
**Рис. 6.** Результат

Вычислим оптимальный h:

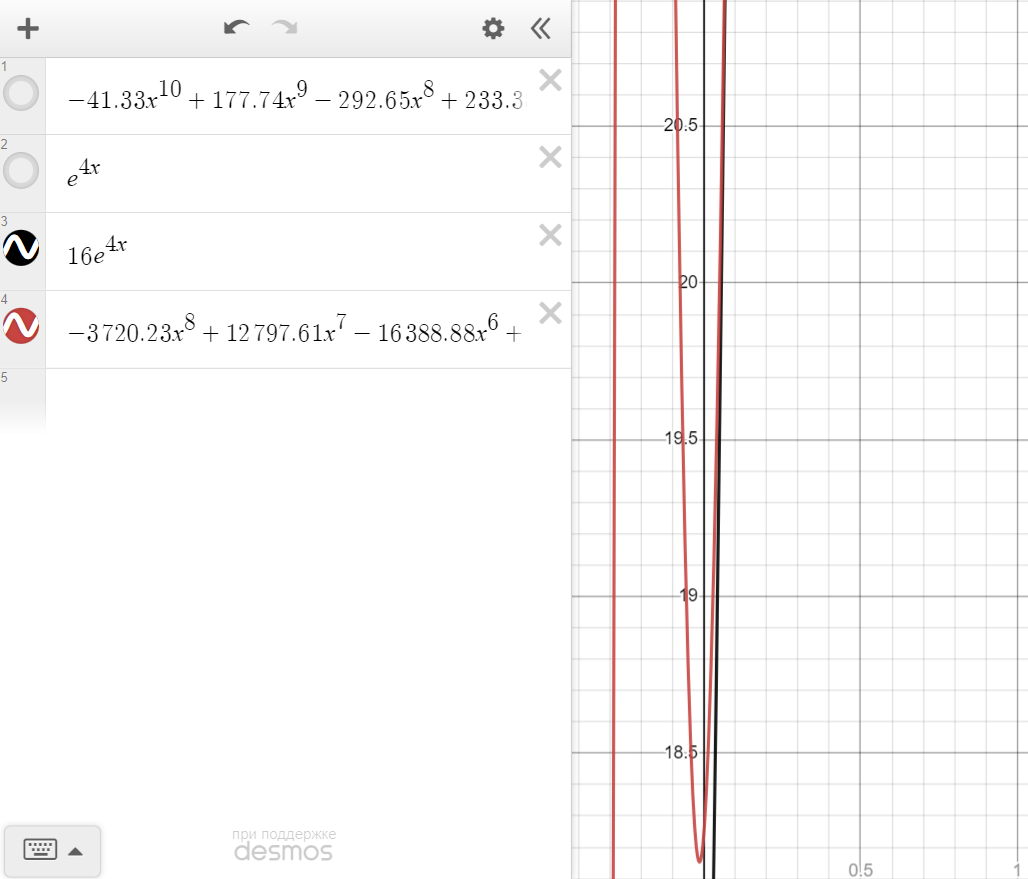
***Задача 3:***



**Рис. 7.** Результат



**Рис. 8.** Полином



**Рис. 9.** Вторая производная

**Вывод:** в ходе выполнения лабораторной работы были получены практические навыки анализа возможностей построения и выделения наиболее важных свойств объектов моделей для моделирования и использования специализированных программных пакетов и библиотек для стандартных вычислений и визуализации результатов численной аппроксимации производных и обоснования выбора алгоритма аппроксимации.

**ПРИЛОЖЕНИЯ**

**Листинг задачи 1:**

import numpy

import math

from prettytable import PrettyTable

def first\_derivative\_oh(func, x: list, n: int):

derivative = n \* [None]

derivative[0] = round((func(x[1]) - func(x[0])) / (x[1] - x[0]))

for i in range(1, n):

derivative[i] = round( (func(x[i]) - func(x[i-1])) / (x[i] - x[i-1]), 3)

return derivative

def first\_derivative\_oh2(func, x: list, n: int):

derivative = n \* [None]

derivative[0] = round( (4\*func(x[1]) - 3\*func(x[0]) - func(x[2])) / (x[1] - x[0]) / 2, 3)

derivative[1] = round( (4\*func(x[2]) - 3\*func(x[1]) - func(x[3])) / (x[2] - x[1]) / 2, 3)

for i in range(2, n):

derivative[i] = round( (3\*func(x[i]) - 4\*func(x[i-1]) + func(x[i-2])) / (x[i] - x[i-1]) / 2, 3)

return derivative

def second\_derivative\_oh2(func, x: list, n: int):

derivative = n \* [None]

for i in range(1, n-1):

derivative[i] = round( (func(x[i+1]) - 2\*func(x[i]) + func(x[i-1])) / (x[i] - x[i-1])\*\*2, 3)

return derivative

def calculate\_inaccuraty(first: list, second: list):

if len(first) != len(second):

return [0]

inacuraty = len(first) \* [0];

for i in range(0, len(first)):

if first[i] is None or second[i] is None:

inacuraty[i] = None

else:

inacuraty[i] = round(abs(first[i] - second[i]), 3)

return inacuraty

def f(x):

return math.e\*\*(4\*x)

def analite\_first\_derivative(x):

return 4 \* math.e\*\*(4\*x)

def analite\_second\_derivative(x):

return 16 \* math.e\*\*(4\*x)

if \_\_name\_\_ == "\_\_main\_\_":

count = 11

x = [round(i, 3) for i in numpy.linspace(0, 1, count)]

y = [round(f(i), 3) for i in x]

first\_derivative = [round(analite\_first\_derivative(i), 3) for i in x]

second\_derivative = [round(analite\_second\_derivative(i), 3) for i in x]

fd1 = first\_derivative\_oh(f, x, len(x))

fd2 = first\_derivative\_oh2(f, x, len(x))

sd2 = second\_derivative\_oh2(f, x, len(x))

table = PrettyTable()

table.add\_column("x", x, "r")

table.add\_column("f(x)", y, "r")

table.add\_column("f'(x)", first\_derivative, "r")

table.add\_column("f~'", fd1, "r")

table.add\_column("O(h)", calculate\_inaccuraty(fd1, first\_derivative), "r")

table.add\_column("f~~'", fd2, "r")

table.add\_column("O(h^2)", calculate\_inaccuraty(fd2, first\_derivative), "r")

table.add\_column("f''(x)", second\_derivative, "r")

table.add\_column("f~''", sd2, "r")

table.add\_column("O(h^2)", calculate\_inaccuraty(sd2, second\_derivative), "r")

print(table)

**Листинг задачи 2:**

from math import \*

def f(x):

return exp(4\*x)

def df(x, h):

return (-3\*f(x) + 4\*f(x + h) - f(x + 2\*h)) / (2\*h)

x = 1

h = 0.1

acc = 4\*exp(4\*x)

print ("h df error")

y = df(x, h)

err = y - acc

while True:

print ("{0:<17} {1:3.5f} {2}".format(h, y, abs(y - acc)))

h = h / 2

y = df(x, h)

if (abs(y - acc) > abs(err)):

print ("{0:<17} {1:3.5f} {2}".format(h, y, abs(y - acc)))

break

err = y - acc

print('h\_эксп.: {0}'.format(2\*h))

**Листинг задачи 3:**

import sympy

import math

import numpy as np

import sympy as sp

from prettytable import PrettyTable

def newton(sym\_x, x, fdd):

n = len(x)

p = fdd[0, 0]

for k in range(1, n):

newton = 1

for m in range(k):

newton \*= (sym\_x - x[m])

p += newton \* fdd[k, 0]

return p

def f(x):

return math.e\*\*(4\*x)

def ddf(x):

return 16 \* math.e\*\*(4\*x)

if \_\_name\_\_ == "\_\_main\_\_":

count = 11

x = [round(i,1) for i in np.linspace(0, 1.0, count)]

y = [round(f(i), 3) for i in x]

table = PrettyTable()

table.add\_column("x", x, "l")

table.add\_column("y", y, "l")

print(table)

n = len(x)

fdd = np.asmatrix(np.zeros((n, n)))

for i in range(n):

fdd[0, i] = y[i]

for j in range(1,n):

for i in range(n-j):

fdd[j, i] = (fdd[j-1, i+1] - fdd[j-1, i]) / (x[i+j] - x[i])

sym\_x = sp.Symbol("x")

p = newton(sym\_x, x, fdd)

p = sp.simplify(p)

print("Newton polynom:", p, '\n')

print("f'':", p.diff(sym\_x, 2), '\n')

x0 = 1

print("xn =", x0)

print("Accurate:", ddf(x0))

print("Newton:", p.diff(sym\_x, 2).subs(sym\_x, x0))

print("Error:", abs(ddf(x0) - p.diff(sym\_x, 2).subs(sym\_x, x0)))