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 $\widetilde{f'}$) и $O(h^2)$ (обозначим $\widetilde{\widetilde{f}'}$) при i = 0, 1, . . . , n.
   б) второй производной функции y = f(x) с порядком погрешности O(h) (обозначим $\widetilde{f''}$) 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.
2) Пользуясь формулой

$$f'(x) = \frac{-3f(x) + 4f(x+h) - f(x+2h)}{2h} + \frac{h^2}{3}f'''(\xi), \xi \in (x, x+2h)$$

   в точке x = 1 вычислить разностную производную второго порядка аппроксимации функции e^{4x}, 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. $\varepsilon = 5 \cdot 10^{-6}$
3) Предполагается заданной таблица значений функции в равноотстоящих узлах x_i, 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 = x_n. Получить выражение для погрешности. Применить формулу для вычисления производной, сравнить с точным значением.

Решение:

Задача 1:

Результаты для исходной функции:

x	f(x)	f'(x)	f~'	O(h)	f~~'	O(h^2)	f''(x)	f~''	O(h^2)
0.0	1.0	4.0	5	1.0	3.709	0.291	16.0	None	None
0.1	1.492	5.967	4.918	1.049	5.533	0.434	23.869	24.189	0.32
0.2	2.226	8.902	7.337	1.565	8.547	0.355	35.609	36.086	0.477
0.3	3.32	13.28	10.946	2.334	12.75	0.53	53.122	53.834	0.712
0.4	4.953	19.812	16.329	3.483	19.021	0.791	79.249	80.311	1.062
0.5	7.389	29.556	24.36	5.196	28.376	1.18	118.225	119.81	1.585
0.6	11.023	44.093	36.341	7.752	42.332	1.761	176.371	178.735	2.364
0.7	16.445	65.779	54.215	11.564	63.151	2.628	263.114	266.641	3.527
0.8	24.533	98.13	80.879	17.251	94.211	3.919	392.52	397.782	5.262
0.9	36.598	146.393	120.657	25.736	140.546	5.847	585.572	593.421	7.849
1.0	54.598	218.393	179.999	38.394	209.67	8.723	873.57	None	None

Рис. 1. $f(x) = e^{4x}$

Для многочлена первой степени приближенные значения совпадают с аналитическими:

x	f(x)	f'(x)	f~'	O(h)	f~~'	O(h^2)	f''(x)	f~''	O(h^2)
0.0	6.0	1	1	0	1.0	0.0	0	None	None
0.1	6.1	1	1.0	0.0	1.0	0.0	0	0.0	0.0
0.2	6.2	1	1.0	0.0	1.0	0.0	0	-0.0	0.0
0.3	6.3	1	1.0	0.0	1.0	0.0	0	0.0	0.0
0.4	6.4	1	1.0	0.0	1.0	0.0	0	-0.0	0.0
0.5	6.5	1	1.0	0.0	1.0	0.0	0	0.0	0.0
0.6	6.6	1	1.0	0.0	1.0	0.0	0	0.0	0.0
0.7	6.7	1	1.0	0.0	1.0	0.0	0	-0.0	0.0
0.8	6.8	1	1.0	0.0	1.0	0.0	0	0.0	0.0
0.9	6.9	1	1.0	0.0	1.0	0.0	0	-0.0	0.0
1.0	7.0	1	1.0	0.0	1.0	0.0	0	None	None

Рис. 2. $f(x) = x + 6$

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

x	f(x)	f'(x)	f~'	O(h)	f~~'	O(h^2)	f''(x)	f~''	O(h^2)
0.0	1.0	1.0	1	0.0	1.0	0.0	2	None	None
0.1	1.11	1.2	1.1	0.1	1.2	0.0	2	2.0	0.0
0.2	1.24	1.4	1.3	0.1	1.4	0.0	2	2.0	0.0
0.3	1.39	1.6	1.5	0.1	1.6	0.0	2	2.0	0.0
0.4	1.56	1.8	1.7	0.1	1.8	0.0	2	2.0	0.0
0.5	1.75	2.0	1.9	0.1	2.0	0.0	2	2.0	0.0
0.6	1.96	2.2	2.1	0.1	2.2	0.0	2	2.0	0.0
0.7	2.19	2.4	2.3	0.1	2.4	0.0	2	2.0	0.0
0.8	2.44	2.6	2.5	0.1	2.6	0.0	2	2.0	0.0
0.9	2.71	2.8	2.7	0.1	2.8	0.0	2	2.0	0.0
1.0	3.0	3.0	2.9	0.1	3.0	0.0	2	None	None

Рис. 3. $f(x) = x^2 + x + 1$

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

x	f(x)	f'(x)	f~'	O(h)	f~~'	O(h^2)	f''(x)	f~''	O(h^2)
0.0	1.0	1.0	1	0.0	0.98	0.02	2.0	None	None
0.1	1.111	1.23	1.11	0.12	1.21	0.02	2.6	2.6	0.0
0.2	1.248	1.52	1.37	0.15	1.5	0.02	3.2	3.2	0.0
0.3	1.417	1.87	1.69	0.18	1.85	0.02	3.8	3.8	0.0
0.4	1.624	2.28	2.07	0.21	2.26	0.02	4.4	4.4	0.0
0.5	1.875	2.75	2.51	0.24	2.73	0.02	5.0	5.0	0.0
0.6	2.176	3.28	3.01	0.27	3.26	0.02	5.6	5.6	0.0
0.7	2.533	3.87	3.57	0.3	3.85	0.02	6.2	6.2	0.0
0.8	2.952	4.52	4.19	0.33	4.5	0.02	6.8	6.8	0.0
0.9	3.439	5.23	4.87	0.36	5.21	0.02	7.4	7.4	0.0
1.0	4.0	6.0	5.61	0.39	5.98	0.02	8.0	None	None

Рис. 4. $f(x) = x^3 + x^2 + x + 1$

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

x	f(x)	f'(x)	f~'	O(h)	f~~'	O(h^2)	f''(x)	f~''	O(h^2)
0.0	0.0	0.0	0	0.0	-0.006	0.006	0.0	None	None
0.1	0.0	0.004	0.001	0.003	-0.01	0.014	0.12	0.14	0.02
0.2	0.002	0.032	0.015	0.017	0.022	0.01	0.48	0.5	0.02
0.3	0.008	0.108	0.065	0.043	0.09	0.018	1.08	1.1	0.02
0.4	0.026	0.256	0.175	0.081	0.23	0.026	1.92	1.94	0.02
0.5	0.062	0.5	0.369	0.131	0.466	0.034	3.0	3.02	0.02
0.6	0.13	0.864	0.671	0.193	0.822	0.042	4.32	4.34	0.02
0.7	0.24	1.372	1.105	0.267	1.322	0.05	5.88	5.9	0.02
0.8	0.41	2.048	1.695	0.353	1.99	0.058	7.68	7.7	0.02
0.9	0.656	2.916	2.465	0.451	2.85	0.066	9.72	9.74	0.02
1.0	1.0	4.0	3.439	0.561	3.926	0.074	12.0	None	None

Рис. 5. $f(x) = x^4$

Задача 2:

```
h                  df        error
0.1                202.49304 15.899564924051958
0.05               215.00005 3.392546139579281
0.025              217.60738 0.7852151507781855
0.0125             218.20362 0.18898073297316387
0.00625            218.34624 0.04636159675919771
0.003125           218.38112 0.011481876436164384
0.0015625          218.38974 0.002857022261878228
0.00078125         218.39189 0.000712582061851208
0.000390625        218.39242 0.00017793663823795214
0.0001953125       218.39256 4.445808622222103e-05
9.765625e-05       218.39259 1.1111663070551003e-05
4.8828125e-05      218.39260 2.777453801172669e-06
2.44140625e-05     218.39260 6.926009064045502e-07
1.220703125e-05    218.39260 1.7178786038130056e-07
6.103515625e-06    218.39260 5.0715925681288354e-08
3.0517578125e-06   218.39260 1.3463022696669213e-08
1.52587890625e-06 218.39260 2.844649316102732e-08
h эксп.: 3.0517578125e-06
```

Рис. 6. Результат

Вычислим оптимальный h:

$$\varepsilon = \frac{h^2}{3} f'''(\xi)$$

$$h^2 = \frac{3\varepsilon}{f'''(\xi)}$$

$$h = \sqrt{\frac{3\varepsilon}{64e^{4x}}}$$

$$h = \sqrt{\frac{3 \cdot 5 \cdot 10^{-6}}{64 \cdot e^4}} \approx 6.55 \cdot 10^{-6}$$

Задача 3:

```
+-----+--------+
| x   | y      |
+-----+--------+
| 0.0 | 1.0    |
| 0.1 | 1.492  |
| 0.2 | 2.226  |
| 0.3 | 3.32   |
| 0.4 | 4.953  |
| 0.5 | 7.389  |
| 0.6 | 11.023 |
| 0.7 | 16.445 |
| 0.8 | 24.533 |
| 0.9 | 36.598 |
| 1.0 | 54.598 |
+-----+--------+
Newton polynom: -41.3359788372723*x**10 + 177.744709000945*x**9
- 292.658730171588*x**8 + 233.382936522764*x**7 - 57.0312500104
899*x**6 - 19.8697916619713*x**5 + 38.0529927235478*x**4 + 2.23
827711662459*x**3 + 9.12546626982115*x**2 + 3.94936904761978*x
+ 1.0

f'': -3720.2380953545*x**8 + 12797.619048068*x**7 - 16388.88888
96089*x**6 + 9802.08333395607*x**5 - 1710.9375003147*x**4 - 397
.395833239426*x**3 + 456.635912682574*x**2 + 13.4296626997475*x
+ 18.2509325396423

xn = 1
Accurate: 873.5704005303077
Newton: 870.558571428491
Error: 3.01182910181649
```

Рис. 7. Результат

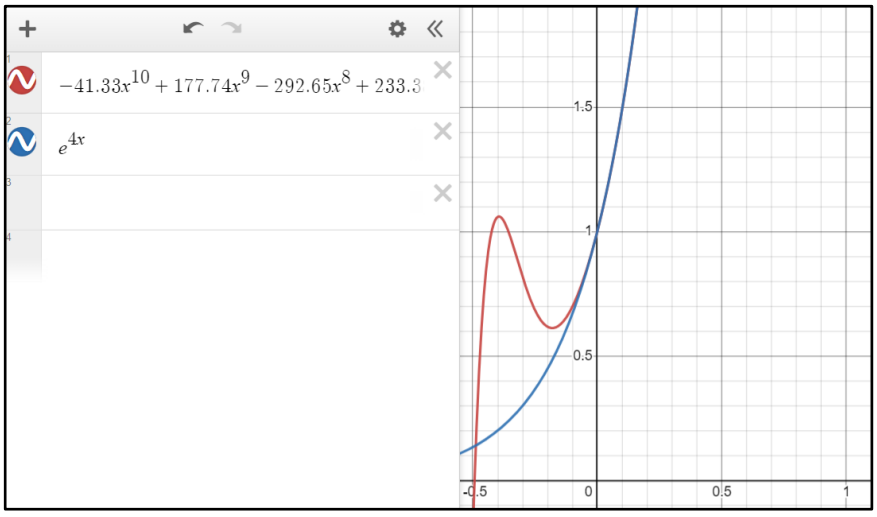

Рис. 8. Полином

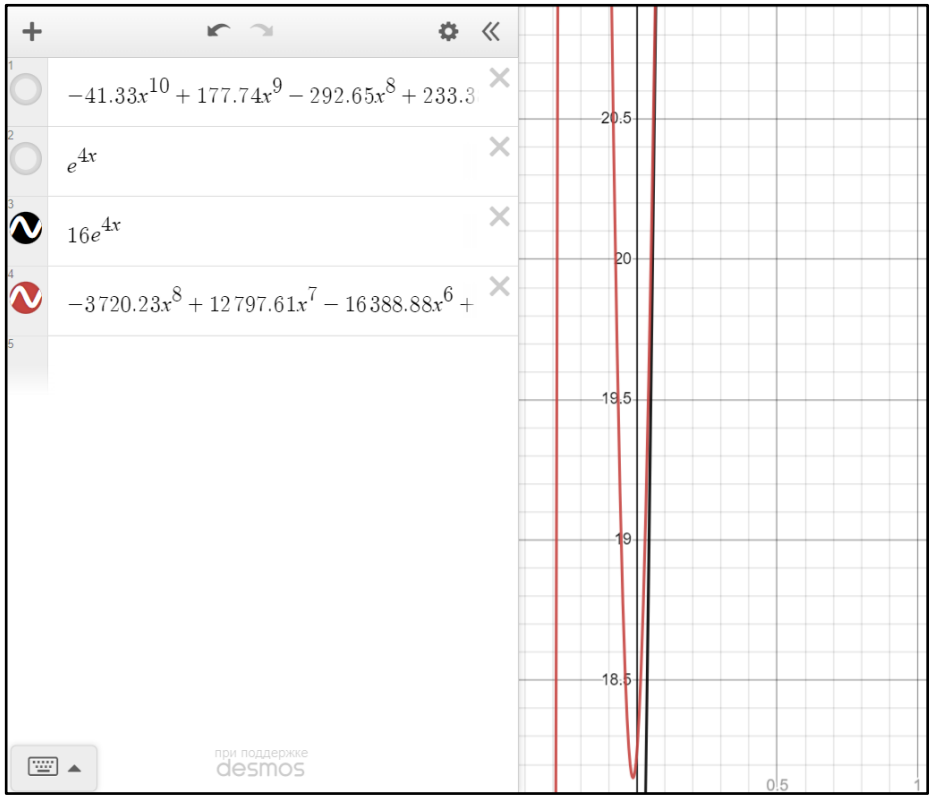

Рис. 9. Вторая производная

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)))
```