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 дефференцирования»

ДИСЦИПЛИНА: «Моделирование»

Выполнил: студент гр. ИУК4-62Б ________________ (Калашников А. С.)
(Подпись) (Ф.И.О.)

Проверил: ________________ (Никитенко У. В.)
(Подпись) (Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:
- Оценка:

Калуга, 2023

Цель работы: 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.

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

Вариант 2

$f(x) = e^{3x}$

№1 Вычислить приближённо значения:

> а) первой производной заданной функции f(x) с порядком погрешности
> $O(h)$ и $O(h^2)$;
>
> б) второй производной заданной функции f(x) с порядком погрешности $O(h^2)$.

Напечатать таблицу значений узлов; значений производных, полученных аналитическим методом в узлах; приближённых значений производных в узлах и фактическую погрешность. Проверить результаты на многочленах соответствующих степеней.

Исходный код программы представлен в Приложении.

Результат выполнения программы для заданной функции $f(x) = e^{3x}$ и диапазона значений от 0,0 до 1,0 с шагом 0,1:

```
i | x | f(x) | f' | f' O(h) | погр. O(h) | f' O(h**2) | погр. O(h**2) | f'' | f'' O(h**2) | погр. O(h**2)
0 | 0.0 | 1.0 | 3.0 | 3 | 0.0 | 2.8866 | 0.1134 | | 9.0 | None | None
1 | 0.1 | 1.3498588075760032 | 4.0496 | 3.4986 | 0.551 | 3.8965 | 0.1531 | | 12.1487 | 12.2401 | 0.0914
2 | 0.2 | 1.822118800390509 | 5.4664 | 4.7226 | 0.7438 | 5.3346 | 0.1318 | | 16.3991 | 16.5224 | 0.1233
3 | 0.3 | 2.4596031111569494 | 7.3788 | 6.3748 | 1.004 | 7.201 | 0.1778 | | 22.1364 | 22.303 | 0.1666
4 | 0.4 | 3.3201169227365477 | 9.9604 | 8.6051 | 1.3553 | 9.7203 | 0.2401 | | 29.8811 | 30.1058 | 0.2247
5 | 0.5 | 4.4816890703380645 | 13.4451 | 11.6157 | 1.8294 | 13.121 | 0.3241 | | 40.3352 | 40.6386 | 0.3034
6 | 0.6 | 6.049647464412945 | 18.1489 | 15.6796 | 2.4693 | 17.7115 | 0.4374 | | 54.4468 | 54.8564 | 0.4096
7 | 0.7 | 8.166169912567646 | 24.4985 | 21.1652 | 3.3333 | 23.908 | 0.5905 | | 73.4955 | 74.0484 | 0.5529
8 | 0.8 | 11.023176380641605 | 33.0695 | 28.5701 | 4.4994 | 32.2725 | 0.797 | | 99.2086 | 99.9549 | 0.7463
9 | 0.9 | 14.879731724872835 | 44.6392 | 38.5656 | 6.0736 | 43.5633 | 1.0759 | | 133.9176 | 134.925 | 1.0074
10 | 1.0 | 20.085536923187664 | 60.2566 | 52.0581 | 8.1985 | 58.8043 | 1.4523 | | 180.7698 | None | None
```

Используем программу с теми же параметрами, но с функцией f(x) = x + 3, у которой f'(x) = 1:

```
i | x | f(x) | f' | f' O(h) | погр. O(h) | f' O(h**2) | погр. O(h**2) | f'' | f'' O(h**2) | погр. O(h**2)
0 | 0.0 | 3.0 | 1 | 1 | 0 | 1.0 | 0.0 | | 0 | None | None
1 | 0.1 | 3.1 | 1 | 1.0 | 0.0 | 1.0 | 0.0 | | 0 | 0.0 | 0.0
2 | 0.2 | 3.2 | 1 | 1.0 | 0.0 | 1.0 | 0.0 | | 0 | -0.0 | 0.0
3 | 0.3 | 3.3 | 1 | 1.0 | 0.0 | 1.0 | 0.0 | | 0 | 0.0 | 0.0
4 | 0.4 | 3.4 | 1 | 1.0 | 0.0 | 1.0 | 0.0 | | 0 | 0.0 | 0.0
5 | 0.5 | 3.5 | 1 | 1.0 | 0.0 | 1.0 | 0.0 | | 0 | 0.0 | 0.0
6 | 0.6 | 3.6 | 1 | 1.0 | 0.0 | 1.0 | 0.0 | | 0 | 0.0 | 0.0
7 | 0.7 | 3.7 | 1 | 1.0 | 0.0 | 1.0 | 0.0 | | 0 | -0.0 | 0.0
8 | 0.8 | 3.8 | 1 | 1.0 | 0.0 | 1.0 | 0.0 | | 0 | 0.0 | 0.0
9 | 0.9 | 3.9 | 1 | 1.0 | 0.0 | 1.0 | 0.0 | | 0 | 0.0 | 0.0
10 | 1.0 | 4.0 | 1 | 1.0 | 0.0 | 1.0 | 0.0 | | 0 | None | None
```

Далее возьмём многочлен второй степени f(x) = x^2 – 3, у которого f'(x) = 2x, а f''(x) = 2:

```
i | x | f(x) | f' | f' O(h) | погр. O(h) | f' O(h**2) | погр. O(h**2) | f'' | f'' O(h**2) | погр. O(h**2)
0 | 0.0 | -3.0 | 0.0 | 0 | 0.0 | -0.0 | 0.0 | | 2 | None | None
1 | 0.1 | -2.99 | 0.2 | 0.1 | 0.1 | 0.2 | 0.0 | | 2 | 2.0 | 0.0
2 | 0.2 | -2.96 | 0.4 | 0.3 | 0.1 | 0.4 | 0.0 | | 2 | 2.0 | 0.0
3 | 0.3 | -2.91 | 0.6 | 0.5 | 0.1 | 0.6 | 0.0 | | 2 | 2.0 | 0.0
4 | 0.4 | -2.84 | 0.8 | 0.7 | 0.1 | 0.8 | 0.0 | | 2 | 2.0 | 0.0
5 | 0.5 | -2.75 | 1.0 | 0.9 | 0.1 | 1.0 | 0.0 | | 2 | 2.0 | 0.0
6 | 0.6 | -2.64 | 1.2 | 1.1 | 0.1 | 1.2 | 0.0 | | 2 | 2.0 | 0.0
7 | 0.7 | -2.5100000000000002 | 1.4 | 1.3 | 0.1 | 1.4 | 0.0 | | 2 | 2.0 | 0.0
8 | 0.8 | -2.36 | 1.6 | 1.5 | 0.1 | 1.6 | 0.0 | | 2 | 2.0 | 0.0
9 | 0.9 | -2.19 | 1.8 | 1.7 | 0.1 | 1.8 | 0.0 | | 2 | 2.0 | 0.0
10 | 1.0 | -2.0 | 2.0 | 1.9 | 0.1 | 2.0 | 0.0 | | 2 | None | None
```

И, наконец, возьмём многочлен третьей степени f(x) = x^3 - 8, у которого f'(x) = $3x^2$, а f''(x) = 6x

```
i | x | f(x) | f' | f' O(h) | погр. O(h) | f' O(h**2) | погр. O(h**2) | f'' | f'' O(h**2) | погр. O(h**2)
0 | 0.0 | -8.0 | 0.0 | 0 | 0.0 | -0.02 | 0.02 | | 0.0 | None | None
1 | 0.1 | -7.999 | 0.03 | 0.01 | 0.02 | 0.01 | 0.02 | | 0.6 | 0.6 | 0.0
2 | 0.2 | -7.992 | 0.12 | 0.07 | 0.05 | 0.1 | 0.02 | | 1.2 | 1.2 | 0.0
3 | 0.3 | -7.973 | 0.27 | 0.19 | 0.08 | 0.25 | 0.02 | | 1.8 | 1.8 | 0.0
4 | 0.4 | -7.936 | 0.48 | 0.37 | 0.11 | 0.46 | 0.02 | | 2.4 | 2.4 | 0.0
5 | 0.5 | -7.875 | 0.75 | 0.61 | 0.14 | 0.73 | 0.02 | | 3.0 | 3.0 | 0.0
6 | 0.6 | -7.784 | 1.08 | 0.91 | 0.17 | 1.06 | 0.02 | | 3.6 | 3.6 | 0.0
7 | 0.7 | -7.657 | 1.47 | 1.27 | 0.2 | 1.45 | 0.02 | | 4.2 | 4.2 | 0.0
8 | 0.8 | -7.4879999999999995 | 1.92 | 1.69 | 0.23 | 1.9 | 0.02 | | 4.8 | 4.8 | 0.0
9 | 0.9 | -7.271 | 2.43 | 2.17 | 0.26 | 2.41 | 0.02 | | 5.4 | 5.4 | 0.0
10 | 1.0 | -7.0 | 3.0 | 2.71 | 0.29 | 2.98 | 0.02 | | 6.0 | None | None
```

Как видим погрешность вычислений появляется только тогда, когда производная является выпуклой функции, например, квадратичной. Это объясняется тем, что при аппроксимации мы принимаем функцию линейной на очень маленьком масштабе, соответственно чем меньше шаг, тем точнее полученное значение.

№2 Пользуясь формулой:

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

в точке x_0=1 вычислить вторую разностную производную второго порядка аппроксимации функции e^{3x}, последовательно уменьшая шаг h до тех пор, пока фактическая погрешность не начнёт возрастать. Определить h оптимальное экспериментально и теоретически, объяснить полученные результаты.

Исходный код программы представлен в Приложении.

Результат выполнения приложения:

```
Задание №2
h                  df        error
0.1                57.97855 2.278058422610023
0.05               59.75009 0.5065244340631025
0.025              60.13705 0.11956485871376543
0.0125             60.22756 0.029053775221719036
0.00625            60.24945 0.007161496131296019
0.003125           60.25483 0.0017777973552668414
0.0015625          60.25617 0.00044288756686938 77
0.00078125         60.25650 0.00011052732549643451
0.000390625        60.25658 2.7607508918947588e-05
0.0001953125       60.25660 6.898805686716969e-06
9.765625e-05       60.25661 1.7244447647613015e-06
4.8828125e-05      60.25661 4.3130700788651666e-07
2.44140625e-05     60.25661 1.0723585575078687e-07
1.220703125e-05    60.25661 2.5890649624216167e-08
6.103515625e-06    60.25661 9.155947111594287e-09
3.0517578125e-06   60.25661 5.372449152218905 5e-09
1.52587890625e-06  60.25661 6.269083030474576e-09
Эксперементальное h: 1.52587890625e-06
Теоретическое h: 0.00016631140197410124
```

$$|R_\varepsilon(x,h,f)| \le 8 * \frac{\xi}{2h} + \frac{h^2}{3}M_3, M_3 = \max|f'''(\xi)|, \xi \in (x, x+2h)$$

Выполним округление до 5-ого знака после запятой, то $\xi = 5 * 10^{-6}$.

Третья производная функции в точке 1 будет равно:

$M_3 = 27 * e^3$

$8 * \frac{\xi}{2h} = \frac{h^2}{3} M_3$

$h^3 = 4\xi * \frac{3}{M_3} = \frac{12\xi}{M_3}$

Подставляя данные значения и учитывая округление до 5-го знака, получаем: $h_{\text{опт}} \approx 0{,}000001$

Вычисленное экспериментально оптимальное значение шага:
$h_{\text{экс}} = 0{,}000002$.

Таким образом, полученное аналитически значение шага примерно равно экспериментальному значению.

№3 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 первой производной с третьим порядком аппроксимации в точке x = x1. Получить выражение для погрешности. Применить формулу для вычисления производной, сравнить с точным значением

Исходный код программы представлен в Приложении

Запишем интерполяционный полином Ньютона с использованием разделённых разностей:

$$P_n(x) = f[x_n] + f[x_n, x_{n-1}](x - x_n) + \dots + f[x_0, \dots, x_n](x - x_0)(x - x_1)$$

$$\dots(x - x_{n-1}) = f[x_n] + \sum f[x_0, x_1, \dots, x_i] \prod_{j=0}^{n} (x - x_j)$$

Вычислим разделённые разности, воспользовавшись формулой:

$$f[x_j, x_{j+1}, \dots, x_{j+i-1}, x_{j+i}] = \frac{f[x_{j+1}, \dots, x_{j+i-1}, x_{j+i}] - f[x_j, x_{j+1}, \dots, x_{j+i-1}]}{x_{j+i} - x_j}$$

```
Разделённые разности:
[[ 298.86740097  967.33959023 1565.48670047 1688.99569781 1366.68669863]
 [ 347.23438048 1123.88826028 1818.83605514 1962.33303753    0.        ]
 [ 403.42879349 1305.77186579 2113.18601077    0.            0.        ]
 [ 468.71738678 1517.09046687    0.            0.            0.        ]
 [ 544.57191013    0.            0.            0.            0.        ]]
```

Для вычисления полинома воспользуемся формулой обратно разделенной разности Ньютона. Продифференцируем:

$$\frac{dP_n}{dx} = s\,\nabla_{P_n} + \frac{(s(s+1))\nabla_{P_n}^2}{2!} + \frac{(s(s+1)(s+2))\nabla_{P_n}^3}{3!} + \cdots,$$

где $s = \frac{x - x_n}{h}$.

$$\frac{dP_n}{dx} = \frac{1}{h}\left(\nabla_{P_n} + \frac{\nabla_{P_n}^2 s(s+1)}{2!} + \frac{\nabla_{P_n}^3 s(s+1)(s+2)}{3!} + \cdots\right)$$

Получаем следующий результат интерполяционного полинома для производной заданной функции:

```
P_5(x)= 5466.74679452802*x**3 - 27323.4876641589*x**2 + 47323.8277990786*x - 27877.4133956282
```

Найдем выражение для погрешности.

Пусть x – случайная точка, отличная от точек $x_0, x_1, \dots, x_n$. Если P_{n+1} – полином интерполяции Ньютона, которые интерполирует функцию $f(x)$ в точках $x_0, x_1, \dots, x_n$ и x, тогда $P_{n+1}(x) = f(x)$. Тогда

$$P_{n+1}(x) = P_n(x) + f[x_0, x_1, \dots, x_n] \prod_{j=0}^{n} (x - x_j)$$

$$\mathrm{f}(x) = P_{n+1}(x) = P(x) + f[x_0, x_1, \dots, x_n] \prod_{j=0}^{n} (x - x_j)$$

$$E_n(x) = f(x) - P_n(x) = f[x_0, x_1, \dots, x_n] \prod_{j=0}^{n} (x - x_j)$$

Продифференцируем и получим:

$$f'(x) - P_n'(x) = \left(f(x) - P_n(x)\right)' = \left(f[x, x_0, x_1, \dots, x_n] \prod_{j=0}^{n} (x - x_j)\right)' =$$

$$= \left(\frac{f^{n+1}\left(\xi(x)\right)}{(n+1)!}(x - x_0) \dots (x - x_n)\right)' =$$

$$= (x - x_1)(x - x_2) \dots (x - x_n) \frac{f^{n+1}\left(\xi(x)\right)}{(n+1)!}$$

Воспользуемся формулой для вычисления первой производной:

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

```
Задание №3
Точки [1.9  1.95 2.   2.05 2.1 ]
Разделённые разности:
[[ 298.86740097  967.33959023 1565.48670047 1688.99569781 1366.68669863]
 [ 347.23438048 1123.88826028 1818.83605514 1962.33303753    0.        ]
 [ 403.42879349 1305.77186579 2113.18601077    0.            0.        ]
 [ 468.71738678 1517.09046687    0.            0.            0.        ]
 [ 544.57191013    0.            0.            0.            0.        ]]
P_5(x)= 5466.74679452802*x**3 - 27323.4876641589*x**2 + 47323.8277990786*x - 27877.4133956282

Первая производнаяпо формуле:           1200.112565255315
Дельта:                                 10.173815222890198
Производная Ньютон):                     1210.2659021176485
Дельта:                                 0.020478360556808184
```

Погрешность при вычислении с помощью полинома Ньютона, имеет погрешность более чем в 1000 раз меньше значения, полученного с помощью формулы численного дифференцирования.

Вывод: в ходе выполнения данной лабораторной работы были сформированы 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.

Приложение

```
import numpy
import sympy
from math import *

def analite_first_derivative(x):
    return 3*exp(3*x)

def analite_second_derivative(x):
    return 9*exp(3*x)
def function(x):
    return exp(3*x)


def divided_diff(x, y):
    n = len(y)
    coef = zeros([n, n])
    coef[:, 0] = y

    for j in range(1, n):
        for i in range(n - j):
            coef[i][j] = \
                (coef[i + 1][j - 1] - coef[i][j - 1]) / (x[i + j] - x[i])

    return coef


def newton_poly_derivative(coef, points, x):
    n = len(points) - 1
    dp = 0
    p = coef[n]

    for k in range(1, n + 1):
        dp *= (x - points[n-k])
        dp += p
        p *= (x - points[n-k])
        p += coef[n-k]

    return dp

def ex3():

    point = 2
    start = 1.9
    stop = 2.1
    n = 5

    points = linspace(start=start, stop=stop, num=n)
    y = [function(x) for x in points]

    newton_divided_diff = divided_diff(points, y)
    coef = newton_divided_diff[0, :]

    newton_first_derivative = newton_poly_derivative(coef, points, 2)
    delta = abs(analite_first_derivative(point) - newton_first_derivative)

    print("Точки",points)
    print(f"Разделённые разности:")
    print(newton_divided_diff)

    sym_x = sympy.Symbol("x")
```

```
    polynom = newton_poly_derivative(coef, points, sym_x)
    polynom = sympy.simplify(polynom)
    print(f"P_{n}(x)=", polynom, '\n')

    print(f"{'Первая производнаяпо формуле:':40} {(4 * function(2.05) - 3 * 
function(2.0) - function(2.1)) / .05 / 2}")
    print(
        f"{'Дельта:':40} {abs(analite_first_derivative(point) - (4 * 
function(2.05) - 3 * function(2.0) - function(2.1)) / .05 / 2)}")
    print(f"{'Производная Ньютон:':40} {newton_first_derivative}")
    print(f"{'Дельта:':40} {abs(analite_first_derivative(point) - 
newton_first_derivative)}")

def ex2():
    x = 1
    h = 0.1
    acc = 3 * exp(3 * x)
    y = df(x, h)
    error = y - acc
    print("h                  df        error")
    while (abs(y - acc) <= abs(error)):
        error = y - acc
        print("{0:<17} {1:3.5f} {2}".format(h, y, abs(error)))
        h = h / 2;
        y = df(x, h)
    print("{0:<17} {1:3.5f} {2}".format(h, y, abs(y - acc)))
    print('Экспереминтальное h: {0}'.format(h))
    print('Теоретическое h: {0}'.format(sqrt(3 * (5 / 1000000) / (27 * exp(3 
* x)))))
def 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]), 4)
    return derivative

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



def analite_first_derivative(x):
    return 3*exp(3*x)

def analite_second_derivative(x):
    return 9*exp(3*x)

if __name__ == "__main__":
    count = 11
    x = [round(i, 3) for i in numpy.linspace(0, 1, count)]
    first_derivative = [round(analite_first_derivative(i), 4) for i in x]
    second_derivative = [round(analite_second_derivative(i), 4) for i in x]

    a1 = derivative_oh(function, x, len(x))
    a2 = derivative_oh2(function, x, len(x))
    a3 = second_derivative_oh2(function, x, len(x))
    inaccuraty = calculate_inaccuraty(a1, first_derivative)
    inaccuraty2 = calculate_inaccuraty(a2, first_derivative)
    inaccuraty3 = calculate_inaccuraty(a3, second_derivative)
    print("i","|","x","|","f(x)","|","f'","|","f' O(h)","|","погр. O(h)","|"
              ,"f' O(h**2)","|","погр. O(h**2)","|","f''","|"
              ,"f'' O(h**2)","|","погр. O(h**2)")
    for i in range(count):

print(str(i),"|",x[i],"|",function(x[i]),"|",first_derivative[i],"|",a1[i],"|
"
              ,inaccuraty[i],"|",a2[i],"|",inaccuraty2[i],"|"
              ,"|",second_derivative[i],"|",a3[i],"|",inaccuraty3[i])
    print("\n")
    print("Задание №2")
    ex2()
    print("\n")
    print("Задание №3")
    ex3()
```