МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт математики и информационных систем

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Дата сдачи на проверку:

«__»__________ 2025 г.

Проверено:

«__»__________2025 г.

Вариант 18

Отчет по лабораторной работе № 1

по дисциплине

«Вычислительная математика»

Разработал студент гр. ИВТб 2302-05-00 ________________ /Соловьев А.С./

(Подпись)

Проверил заведующий кафедры ЭВМ ________________ /Старостин П.А./

(Подпись)

Работа защищена «__»____________2025 г.

Киров 2025

1 Задание

1. Построить график функции f(x) и отделить один из корней уравнения f.(x)

2. Сузить интервал изоляции корня, если необходимо проверить условие: $M < 2m$

3. Уточнить корень с погрешностью e<= 0.00001 двумя численными методами: комбинированный методом и методом итераций.

4. Проверить полученное знаечние, используя систему Mathcad.

5. Сделать для $y = 3x - e^x = 0, x \in [1, 1.9]$

2 Теорическая часть.Описание методов решения уравнений

Рассмотрим исходный график. В нем функция возврастает до точки $ln(3)$. Корень находится на пересении осью OY ось OX. Пересечени находится в точке значение, которой примерно равно 1.5. Поэтому сузим график справа и слева. Теперь новый график будет на отрезке $x \in [1.4, 1.6]$

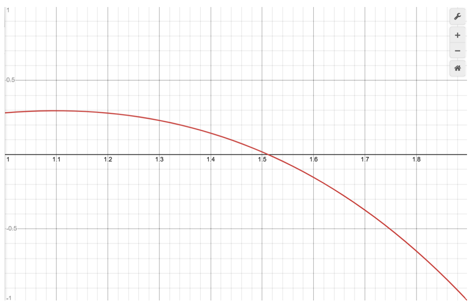

Рис. 1: График исходной функции

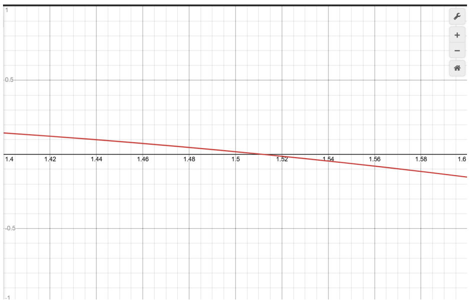

Рис. 2: График на новом интервале

В данной работе рассматриваются численные методы решения нелинейного уравнения $y = 3x - e^x$. Для нахождения корней используются следующие методы:

1. **Метод Ньютона (метод касательных)**

   - Итерационная формула:
     $x_{n+1} = x_n - \frac{f(x)}{f\prime(x)}$
   - Перед применение метода нужно проверить условие $M < 2m$.
     $M = max(\mid f\prime\prime(x) \mid)$
     $m = min(\mid f\prime(x) \mid)$

2. **Метод Хорд**

   - Итерационная формула $x_{n+1} = x_n - f(x_n) \cdot \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n+1})}$
   - Перед применение метода нужно проверить условие $M < 2m$.
     $M = max(\mid f\prime\prime(x) \mid)$
     $m = min(\mid f\prime(x) \mid)$
   - Для работы данного метода нужно чтобы ни одна точка не была ни критической, ни стационарной. То есть $f\prime(x) \neq 0$ и $f\prime\prime(x) \neq 0$.

3. **Метод простых итераций**

   - приведение уравнения к эквивалентной форме $x_{n+1} = \varphi(x_n)$

- Метод требует выбора $\varphi(x)$ удовлетворяющего условия сходимости

4. **Комбинированный метод** Комбинационный метод состоит из следующих этапов:

   - Проверка условия $M < 2m$
   - Итерация метода Хорд
   - Итерация метода Ньютона
   - Алгоритм повторяет, пока корень не будет вычислен с нужной погрешностью

Ограничения методов

- Метод Ньютона может не сойтись при плохом выборе начального приближения.
- Метод хорд требует выбора двух начальных приближений и проверки на стационарные и критические точки.
- Метод простых итераций требует правильного выбора итерационной функции.

3 Практическая часть

Анализ условий для работы методов Комбинированный метод

1. **Метод хорд**

   Для работы метода хорд нужно, чтобы ни одна точка на отрезке $x \in [1.4, 1.6]$ не была стационарной или критической и должно соблюдаться условие $M < 2m$. При $x \in [1.4, 1.6]$.

Теперь проверим условие $M < 2m$ на новом отрезке.

$m = min(|f\prime(x)|)$

$M = max(|f\prime\prime(x)|)$

При $x_0 = 1.5$ и $x_1 = 1.55$

В данном случае $m =\approx 1.4816890703380645$, а $M \approx 4.71147018259074$. Отсюда следует, что условие

$4.71147018259074 <= 2.963378140676129$ не выполнено. Поэтому будем использовать дуругю формулу для уточнения приближения к корню, о ней будет написано дальше.

Проверим другие условия:

$f\prime(x) \neq 0$ и $f\prime\prime(x) \neq 0$

$f\prime(x) = (3 - e^x)$

$(3 - e^x) \neq 0$

$e^x \neq 3 \Rightarrow$x $\neq \ln 3$

$ln 3 \approx 1,0986$

$1.4 > ln(3)$, значит данное условие выполняется на заданном отрезке $x \in [1.4, 1.6]$ Проверим второе условие $f\prime\prime(x) \neq 0$

$f\prime\prime(x) = e^x$

Отсюда следует, что данное условие $x \neq 0$ у нас соблюдено.

Значит, все условия для работы метода хорд соблюдены.

2. **Метод Ньютона (метод касательных)**

Как я упоминал выше, чтобы работал метод Ньютона должно соблюдаться условие $M < 2m$. Проверим данное условие.

$m = min(|f\prime(x)|)$

$M = max(|f\prime\prime(x)|)$ При $x_0 = 1.5$ и $x_1 = 1.55$

В данном случае $m =\approx 1.4816890703380645$, а $M \approx 4.71147018259074$. Отсюда следует, что условие

$4.71147018259074 <= 2.963378140676129$ не выполнено, даже при очень большом приближении. Так как функция монотонно возврастающая и это экспонента. У нее не может быть такого прироста, чтобы условие выполнялось. Поэтому будем использовать другую формулу для слежения за точностью приближения к корню по общей более сложной формуле:

$|\varepsilon - x_n| <= \frac{f(x_n)}{min(f\prime(x))}$

$|\varepsilon - x_n|$ - это погрешность с которой нужно посчитать, поэтому можем заменить $|\varepsilon - x_n|$ на ε , которое дано было по заданию (10^{-5}).

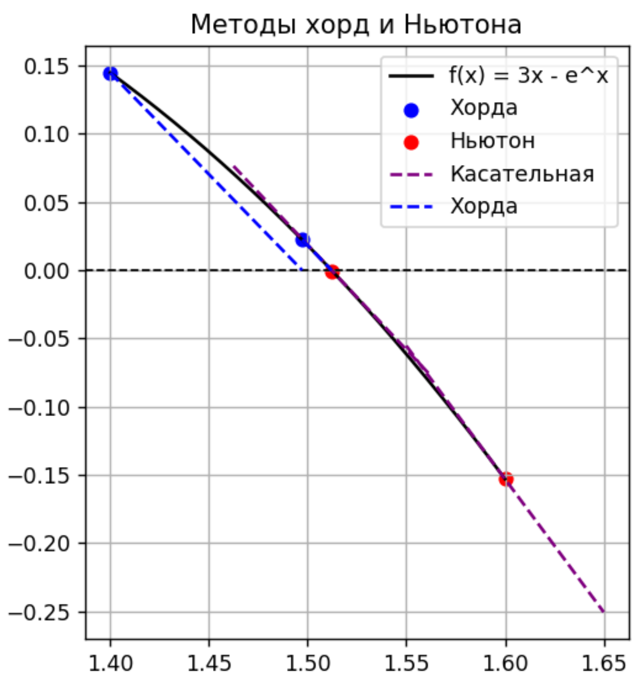

Рис. 4: График функции применения комбинационного метода

Метод Хорд будет применяться слевой стороны,то есть точка справа будет неподвижной. Метод Ньютона будет применяться с правой стороны, он будет двигаться к корню.

Комбинированный метод будет повторяться пока выполнимо условие
$10^{-5} <= \frac{f(x_n)}{min(f\prime(x))}$.

Итерационный метод

Для работы данного метода функция эквивалентная $\varphi(x)$ должна быть непрерывна на отрезке.

Эквивалентной функцией для $y = 3x - e^x$ будет $\varphi(x) = \ln(3x)$. Функция $\varphi(x)$ сходится во всех случаях, кроме $x = 0$, но данное значение не пренадлежит нашему промежутку, значит наш отрезок $x \in [1.4, 1.6]$ удовлетворяет условию сходимости.

Условие прекращения итерирования: $|x1 - x0| < 10^{-5}$. Где $x1$ - граница спарва от корня, $x0$ - граница слева от корня.

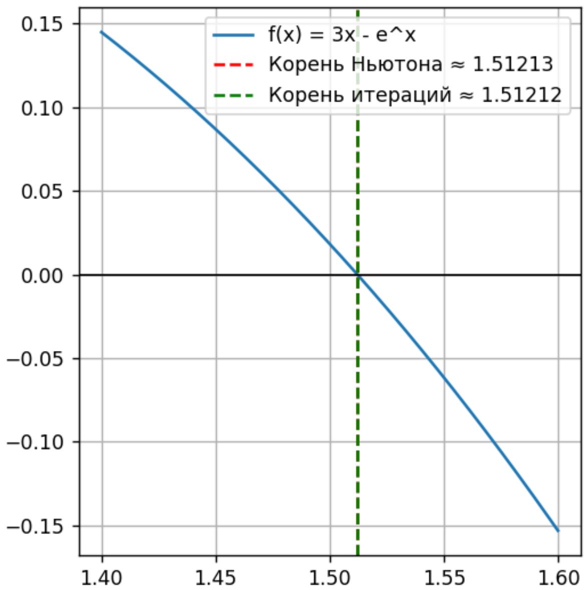

Рис. 5: График с корнями

Листинг программы:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


# Определение функции и ећ производной
def f(x):
    return 3 * x - np.exp(x)


```

```
def df(x):
    return 3 - np.exp(x)


def ddf(x):
    return (np.exp(x)) * (-1)


# Проверка условия сходимости
def check_convergence(a, b):
    m = min(abs(df(a)), abs(df(b)))  # Берем минимум по модулю
    M = max(abs(ddf(a)), abs(ddf(b)))  # В идеале, найти максимум на всем интервале

    print(f"a: {a}, b: {b}")
    print(f"df(a): {df(a)}, df(b): {df(b)}")
    print(f"ddf(a): {ddf(a)}, ddf(b): {ddf(b)}")
    print(f"m = {m}, M = {M}, Условие M <= 2m: {M <= 2 * m}")

    return M <= 2 * m, m, M

# Метод Ньютона
def newton_method(x0):
    return x0 - f(x0) / df(x0)


# Метод хорд
def chord_method(x0, x1):
    return x1 - f(x1) * (x1 - x0) / (f(x1) - f(x0))


# Метод простых итераций
def iteration_method(x0, tol=1e-5, max_iter=26):
    def phi(x):
        return np.log(3 * x)  # Итерационная функция

    arr_iteration = []
    x = x0
    for _ in range(max_iter):
        x_new = phi(x)
```

```
        arr_iteration.append(x_new)
        if abs(x_new - x) < tol:
            return arr_iteration
        x = x_new
    return arr_iteration


def combined_method(x0, x1, tol=1e-5):
    iteration_newton = []
    iteration_chord = []
    conv, m, M = check_convergence(x0, x1)

    iteration_chord.append(x0)
    iteration_newton.append(x1)
  #  print("Условие M < 2m выполнено")

    while abs(f(x1)) / m > tol:  # Новый критерий остановки
        x0 = chord_method(x0, x1)
        x1 = newton_method(x0)
        iteration_chord.append(x0)
        iteration_newton.append(x1)
    return iteration_newton, iteration_chord, m, M


# Вызываем функции и получаем корни для разных методов
root_newton, root_chord, m, M = combined_method(x0=1.4, x1=1.6, tol=1e-5)
root_iteration = iteration_method(x0=1.1, tol=1e-5)

if root_newton:
    print(f"Приближенное решение (комбинированный метод): {root_newton[-1]}")
    print(root_newton)
    print(root_chord)
if root_iteration:
    print(f"Приближенное решение (метод простых итераций): {root_iteration[-1]}
       ")
print(f"m = {m}, M = {M}")

# Вывод таблицы для метода итераций
if root_iteration:
    data_iteration = pd.DataFrame({
```

```
        "Значение функции": [f(x) for x in root_iteration],
        "Значение корня": root_iteration
    })
    print(data_iteration)

# Объединяем итерации хорд и Ньютона

if root_chord:
    data_combination = pd.DataFrame({
        "Значение корня методом хорд": [f(x) for x in root_chord],
        "Значение методом Ньютона": [f(x) for x in root_newton],
        "Значение корня": root_chord
    })
    print(data_combination)

# График функции
x_range = np.linspace(1.4, 1.6, 400)
y_range = f(x_range)

plt.figure(figsize=(10, 5))

# График метода хорд и Ньютона
plt.subplot(1, 2, 1)
plt.plot(x_range, y_range, label="f(x) = 3x - e^x", color='black')
plt.axhline(0, color="black", linewidth=1, linestyle="--")
for i in range(min(2, len(root_chord))):
    x_c = root_chord[i]
    x_n = root_newton[i]

    plt.scatter(x_c, f(x_c), color='blue', label="Хорда" if i == 0 else "")
    plt.scatter(x_n, f(x_n), color='red', label="Ньютон" if i == 0 else "")

    # Правильная касательная (линию строим в небольшом диапазоне вокруг x_n)
    tangent_x = np.linspace(x_n - 0.05, x_n + 0.05, 100)
    tangent_y = f(x_n) + df(x_n) * (tangent_x - x_n)
    plt.plot(tangent_x, tangent_y, color='purple', linestyle='--', label="Касат
        ельная" if i == 0 else "")

    # Хорда
    plt.plot([x_c, root_chord[i + 1]], [f(x_c), 0], color='blue', linestyle='--
```

```
', label="Хорда" if i == 0 else "")

plt.legend()
plt.title("Методы хорд и Ньютона")
plt.grid()

# График корней
plt.subplot(1, 2, 2)
plt.plot(x_range, y_range, label="f(x) = 3x - e^x")
plt.axhline(0, color="black", linewidth=1)
if root_newton:
    plt.axvline(root_newton[-1], color="red", linestyle="--", label=f"Корень Ньютона {root_newton[-1]:.5f}")
if root_iteration:
    plt.axvline(root_iteration[-1], color="green", linestyle="--", label=f"Корень итераций {root_iteration[-1]:.5f}")
plt.legend()
plt.grid()
plt.show()
```

Листинг 1: Python код программы

Итерации методов

	Значение корня методом хорд	Значение методом Ньютона	Значение корня
0	1.448000e-01	-1.530324e-01	1.400000
1	2.238936e-02	-1.481562e-02	1.497236
2	3.232158e-04	-2.007795e-04	1.511924
3	6.236972e-08	-3.870907e-08	1.512135

Рис. 6: Итерации комбинированного метода

Для вычисления корня с нужной точностью было сделано 2 итерации. Корень отмечен на Рис.4 красной горизонтальной пунктирной линией.

	Итерация	Значение функции	Значение корня
0	1	0.295837	1.098612
1	2	0.282143	1.192660
2	3	0.246415	1.274798
3	4	0.199806	1.341400
4	5	0.152778	1.392326
5	6	0.111786	1.429588
6	7	0.079231	1.455999
7	8	0.054917	1.474304
8	9	0.037482	1.486799
9	10	0.025317	1.495237
10	11	0.016980	1.500897
11	12	0.011334	1.504675
12	13	0.007542	1.507190
13	14	0.005008	1.508859
14	15	0.003321	1.509966
15	16	0.002200	1.510699
16	17	0.001457	1.511185
17	18	0.000964	1.511506
18	19	0.000638	1.511719
19	20	0.000422	1.511860
20	21	0.000279	1.511953
21	22	0.000185	1.512014
22	23	0.000122	1.512055
23	24	0.000081	1.512082
24	25	0.000053	1.512100
25	26	0.000035	1.512112

Рис. 7: Итерации метода простых итераций

Для того, чтобы достить корня с нужно погрешностью понадобилось 26 итераций. Корень отмечен на Рис.4 зеленой горизонтальной пунктирной линией.

4 Проверка в вычислений в Maxima

Сделаем проверку значения корня в Maxima

```
/* Определяем уравнение */
expr: 3·x − exp(x);

/* Находим корень на интервале [1.4, 1.6] */
root: find_root(expr, x, 1.4, 1.6);

/* Выводим корень */
print("Корень уравнения: ", root);

/* Строим график */
plot2d(expr, [x, 1.4, 1.6]);
```

$3\,x - \%e^{x}$

1.5121345516578424

Корень уравнения: 1.5121345516578424

1.5121345516578424

Рис. 8: График функции

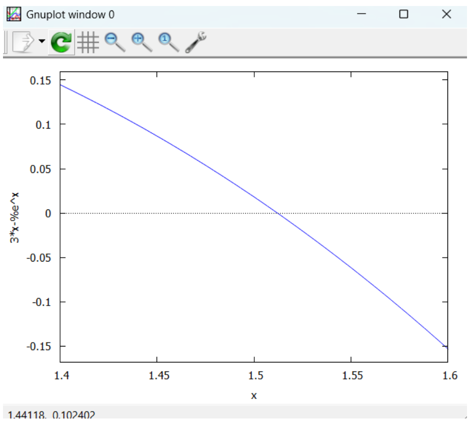

Рис. 9: Вычисления в maxima

Значение корня вычесленное с помощью Maxima получилось 1.51213 ± 0.00001 , что примерно равно вычислениям в программе python, в ней корень равен 1.51212 ± 0.00001 методом итераций, а методом Ньютона корень равен 1.51213 ± 0.00001.

5 Вывод

В ходе данной лабораторной работы, я смог ознакомлся с 3 методами решения нелинейных уравнений. А именно: метод Ньютона, метод хорд, метод простых итераций. Эти методы были реализованы в программес использованием высокого языка программирования (Python). Также вычесленные корни были проверены в Maxima, они сошлись с минимальной погрешностью.