**ОТЧЁТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1**

**ПРИБЛИЖЕННЫЕ МЕТОДЫ РЕШЕНИЯ**

**НЕЛИНЕЙНЫХ УРАВНЕНИЙ**

**(Вариант 9)**

*Выполнил студент 3 курса ПМиИ*

*Кондратьев Виталий*

***Постановка задачи:*** Исследовать функцию и решить уравнение .

I. Найти промежуток, содержащий наименьший положительный корень уравнения, для которого выполняются достаточные условия сходимости одного из итерационных методов;

II. Получить приближенное решение (с точностью 10-7) методами:

1) *методом Ньютона (метод касательных)*

;

2) *методом хорд*

;

3) *методом секущих*

;

4) *конечноразностным методом Ньютона*

 — малый параметр;

5) *методом Стеффенсена*

;

6) *методом простых итераций*



Если , то .

Для оценки погрешности приближенного решения, полученного любым методом, может использоваться неравенство

.

***Результаты расчетов***

a = 0,5; b = 5,5; n = 10;

Таблица значений функции (см. программу 1 в приложении)

|  |  |
| --- | --- |
|  |  |
| 0.5000000 | -3.3862944 |
| 1.0000000 | -1.0000000 |
| 1.5000000 | 0.1442635 |
| 2.0000000 | 0.8862944 |
| 2.5000000 | 1.4325815 |
| 3.0000000 | 1.8638912 |
| 3.5000000 | 2.2198117 |
| 4.0000000 | 2.5225887 |
| 4.5000000 | 2.7859326 |
| 5.0000000 | 1.5694379 |

# График функции

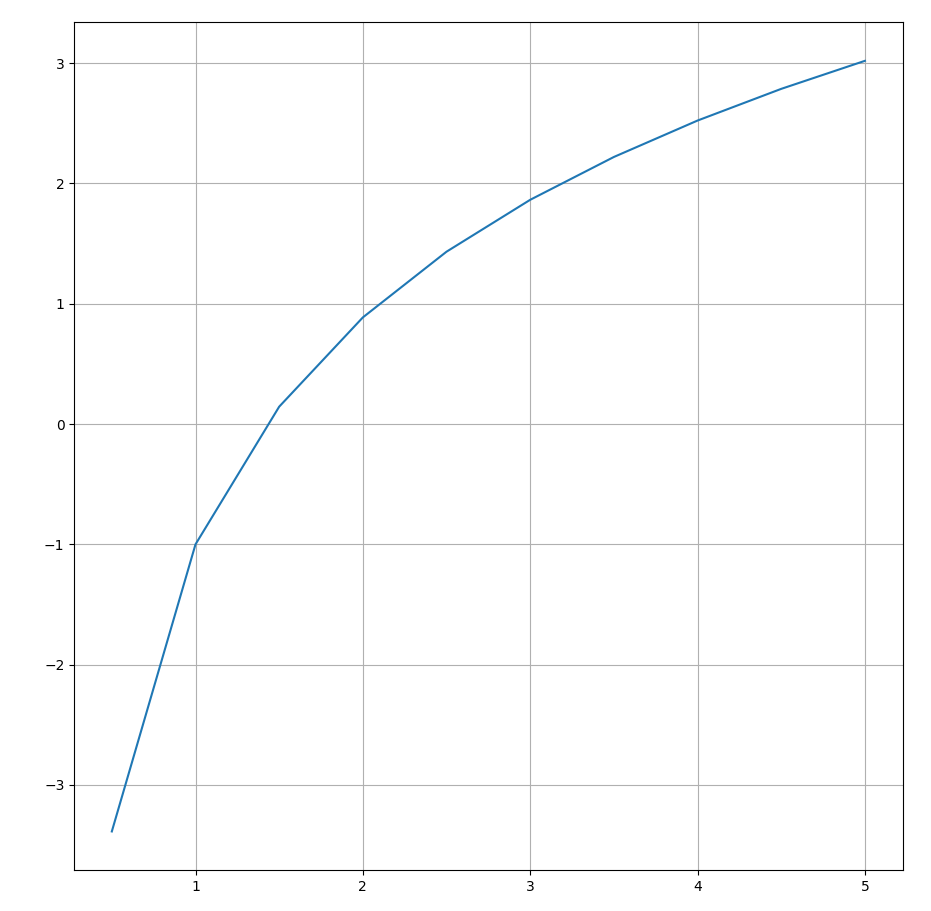


Таблица и график сделаны с помощью ЯП Python.

Построив график функции, определяем, что уравнение имеет только один корень, который находится в интервале .

Уточним значение корня с требуемой точностью 10-7, пользуясь методами 1–6.

**Метод Ньютона (метод касательных).**

Определим поведение первой и второй производных функции на интервале .

Для функции

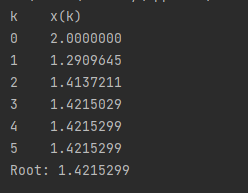
График производных выглядит следующим образом:



Видим, что вторая производная отрицательна по началу определения функции (до точки корня, потом совпадает со знаком первой производной) , поэтому в качестве начального приближения можно взять правую границу интервала, т.е. . Приближенные значения вычисляются по формуле:

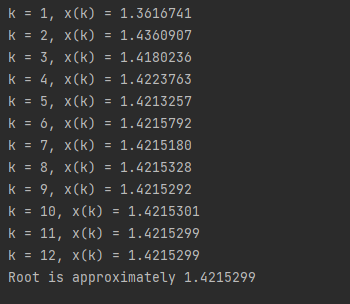
.

Итерации завершаются при выполнении условия . (в данном случае если взять левую границу – результат не изменится)

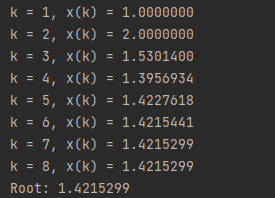


**Метод хорд.** Вычисления проводятся по формуле . Итерации завершаются при выполнении условия .

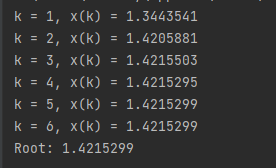
Начальное приближение .



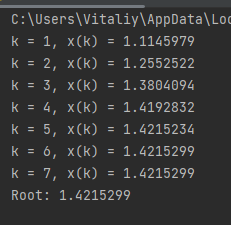
**Метод секущих.** В качестве начальных точек зададим: и . Дальнейшие вычисления проводятся по формуле . Итерации завершаются при выполнении условия .



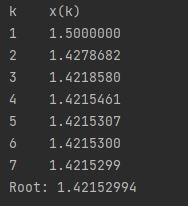
**Конечноразностный метод Ньютона.** В качестве начального приближения берем . Выбираем параметр . Вычисления проводятся по формуле .



**Метод Стеффенсена.** В качестве начального приближения берем Вычисления проводятся по формуле .



**Метод простых итераций.** Выбираем . Вычисления проводятся по формуле . Выбираем τ=0.5, удовлетворяющее условию .



**Итоговая таблица**

|  |  |  |  |
| --- | --- | --- | --- |
| Метод решения | Выбранный интервал | Полученное решение | Количество итераций |
| 1. Метод Ньютона (метод касательных) |  | 1.4215299 | 6 |
| 2. Метод хорд |  | 1.4215299 | 12 |
| 3. Метод секущих |  | 1.4215299 | 8 |
| 4. Конечноразностный метод Ньютона |  | 1.4215299 | 6 |
| 5. Метод Стеффенсена |  | 1.4215299 | 7 |
| 6. Метод простых итераций |  | 1.4215299 | 7 |

**Вывод:**

Исходя полученных результатов, можно сделать вывод, что метод касательных Ньютона и конечно-разностный метод Ньютона были наиболее эффективными в плане количества итераций. Оба метода сходились к решению за 6 итераций.

Однако, если говорить о точности методов, то стоит упомянуть, что метод касательных Ньютона обычно считается более точным, чем конечно-разностный метод Ньютона. Это связано с тем, что метод касательных использует информацию о производной функции в каждой точке, тогда как конечно-разностный метод приближает производную конечной разностью.

Приближенным решением уравнения является

**ПРИЛОЖЕНИЕ**

***Программа построения таблицы значений функции***

*"""  
Лабораторная работа №1  
Тема: ПРИБЛИЖЕННЫЕ МЕТОДЫ РЕШЕНИЯ НЕЛИНЕЙНЫХ УРАВНЕНИЙ  
Студент ОНК «ИВТ» ВШ КНиИИ направления ПМиИ 3 курса  
Кондратьев Виталий  
Вариант 9  
"""*import numpy as np  
import matplotlib.pyplot as plt  
import math  
  
"""I этап"""  
a = 0.5 # int(input('Введите левую границу: '))  
b = 5.5 # int(input('Введите правую границу: '))  
n = 10 # int(input('Укажите число точек: '))  
  
h = (b - a) / n # шаг  
x = np.arange(a, b, h) # равномерно расположенные значения внутри заданного интервала  
x = np.around(x, 3) # округляем  
# x = np.linspace(a, b, n) # с произвольным шагом  
  
print(x)  
  
y = 2\*np.log(x) - 1/x # решение f(x)  
y = np.around(y, 7)  
plt.subplot(122)  
columns = ['x', 'f(x)']  
list\_of\_values = np.array([x, y]).T  
plt.table(cellText=list\_of\_values, colLabels=columns, loc='upper center')  
plt.axis('off')  
  
plt.subplot(121)  
plt.tight\_layout()  
print(y)  
  
plt.plot(x, y)  
plt.grid()  
plt.show()

***Программы нахождения корня всеми способами***

import math  
  
def f(x):  
 return 2\*math.log(x) - 1/x  
  
def df(x):  
 return (2\*x+1)/(x\*\*2)  
  
def newton\_method(x0, eps):  
 *"""Метод Ньютона (Метод косательных)"""* k = 0  
 x = x0  
 print("k\t x(k)")  
 print("{}\t {:.7f}".format(k, x))  
 while True:  
 k += 1  
 x\_new = x - f(x) / df(x)  
 print("{}\t {:.7f}".format(k, x\_new))  
 if abs(x\_new - x) < eps:  
 break  
 x = x\_new  
 return x\_new  
  
eps = 1e-7  
x = newton\_method(2, eps)  
  
print("Root: {:.7f}".format(x))

import math  
  
def f(x):  
 return 2\*math.log(x) - 1/x  
  
def chord\_method(a, b, eps):  
 *"""Метод хорд"""* x0 = a - ((b - a) \* f(b)) / (f(b) - f(a))  
 x1 = x0 - (f(x0) \* (b - x0)) / (f(b) - f(x0))  
 k = 1  
 while abs(x1 - x0) > eps:  
 x0, x1 = x1, x1 - (f(x1) \* (b - x1)) / (f(b) - f(x1))  
 print("k = {}, x(k) = {:.7f}".format(k, x1))  
 k += 1  
 return x1  
  
a = 1  
b = 2  
eps = 1e-7  
  
root = chord\_method(a, b, eps)  
  
print("Root is approximately {:.7f}".format(root))

import math  
  
def f(x):  
 return 2\*math.log(x) - 1/x  
  
def secant\_method(x0, x1, eps):  
 *"""Метод секущих"""* k = 1  
 print(f'k = {k}, x(k) = {x0:.7f}')  
 k += 1  
 print(f'k = {k}, x(k) = {x1:.7f}')  
 while True:  
 x = x1 - (f(x1) / (f(x1) - f(x0))) \* (x1 - x0)  
 k += 1  
 print(f'k = {k}, x(k) = {x:.7f}')  
 if abs(x - x1) < eps:  
 return x  
 x0, x1 = x1, x  
  
eps = 1e-7  
x0 = 1  
x1 = 2  
  
root = secant\_method(x0, x1, eps)  
print(f'Root: {root:.7f}')

import math  
  
def f(x):  
 return 2\*math.log(x) - 1/x  
  
def newton\_fd\_method(x0, h, eps):  
 *"""Конечно-разностный метод Ньютона"""* k = 0  
 while True:  
 fx = f(x0)  
 fxh = f(x0 + h)  
 x = x0 - (h \* fx) / (fxh - fx)  
 k += 1  
 print(f'k = {k}, x(k) = {x:.7f}')  
 if abs(x - x0) < eps:  
 return x  
 x0 = x  
  
x0 = 1  
h = 0.05  
eps = 1e-7  
  
root = newton\_fd\_method(x0, h, eps)  
print(f'Root: {root:.7f}')

import math  
  
def f(x):  
 return 2\*math.log(x) - 1/x  
  
def steffensen\_method(x0, eps):  
 k = 0  
 while True:  
 x1 = x0 - (f(x0)\*\*2) / (f(x0 + f(x0)) - f(x0))  
 k += 1  
 print(f'k = {k}, x(k) = {x1:.7f}')  
 if abs(x1 - x0) < eps:  
 return x1  
 x0 = x1  
  
x0 = 2  
eps = 1e-7  
  
root = steffensen\_method(x0, eps)  
print(f'Root: {root:.7f}')

import math  
  
def f(x):  
 return 2\*math.log(x) - 1/x  
  
def simple\_iterations\_method(a, b, eps):  
 *"""Метод простых итераций"""* t = 1 / (2 \* max(abs(f(a)), abs(f(b))))  
 x = a  
 k = 0  
 print(f'k \t x(k)')  
 while True:  
 x\_next = x - t \* f(x)  
 k += 1  
 print(f'{k} \t {x\_next:.7f}')  
 if abs(x\_next - x) < eps:  
 return x\_next  
 x = x\_next  
  
a = 1  
b = 2  
eps = 1e-7  
  
root = simple\_iterations\_method(a, b, eps)  
print(f'Root: {root:.8f}')