# ОТЧЁТ
# ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1

# ПРИБЛИЖЕННЫЕ МЕТОДЫ РЕШЕНИЯ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

**(Вариант 9)**

*Выполнил студент 3 курса ПМиИ*
*Кондратьев Виталий*

***Постановка задачи:*** Исследовать функцию $f(x) = 2\ln x - \frac{1}{x}$ и решить уравнение $f(x) = 0$.

I. Найти промежуток, содержащий наименьший положительный корень уравнения $f(x) = 0$, для которого выполняются достаточные условия сходимости одного из итерационных методов;

II. Получить приближенное решение (с точностью $10^{-7}$) методами:

1) *методом Ньютона (метод касательных)*

$$x_0 = a, \quad x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad f(a)f''(a) > 0;$$

2) *методом хорд*

$$x_0 = a, \quad x_{k+1} = x_k - \frac{f(x_k)}{f(b) - f(x_k)}(b - x_k), \quad f(b)f''(b) > 0;$$

3) *методом секущих*

$$x_0, x_1 \in [a,b], \quad x_{k+1} = x_k - \frac{f(x_k)}{f(x_k) - f(x_{k-1})}(x_k - x_{k-1});$$

4) *конечноразностным методом Ньютона*

$$x_0 \in [a,b], \quad x_{k+1} = x_k - \frac{h \cdot f(x_k)}{f(x_k + h) - f(x_k)}, \quad h > 0 \text{ — малый параметр;}$$

5) *методом Стеффенсена*

$$x_0 \in [a,b], \quad x_{k+1} = x_k - \frac{f^2(x_k)}{f\left(x_k + f(x_k)\right) - f(x_k)};$$

6) *методом простых итераций*

$$x_0 \in [a,b], \quad x_{k+1} = x_k - \tau f(x_k),$$

Если $f'(x) > 0$, то $0 < \tau < \dfrac{2}{\min\left(f'(x)\right)}$.

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

$$\left|x_k - x^*\right| < \frac{\left|f(x_k)\right|}{m}, \quad m = \min_{[a,b]}\left|f'(x)\right|.$$

***Результаты расчетов***

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

Таблица значений функции (см. программу 1 в приложении)

| $x$ | $f(x)$ |
|---|---|
| 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**

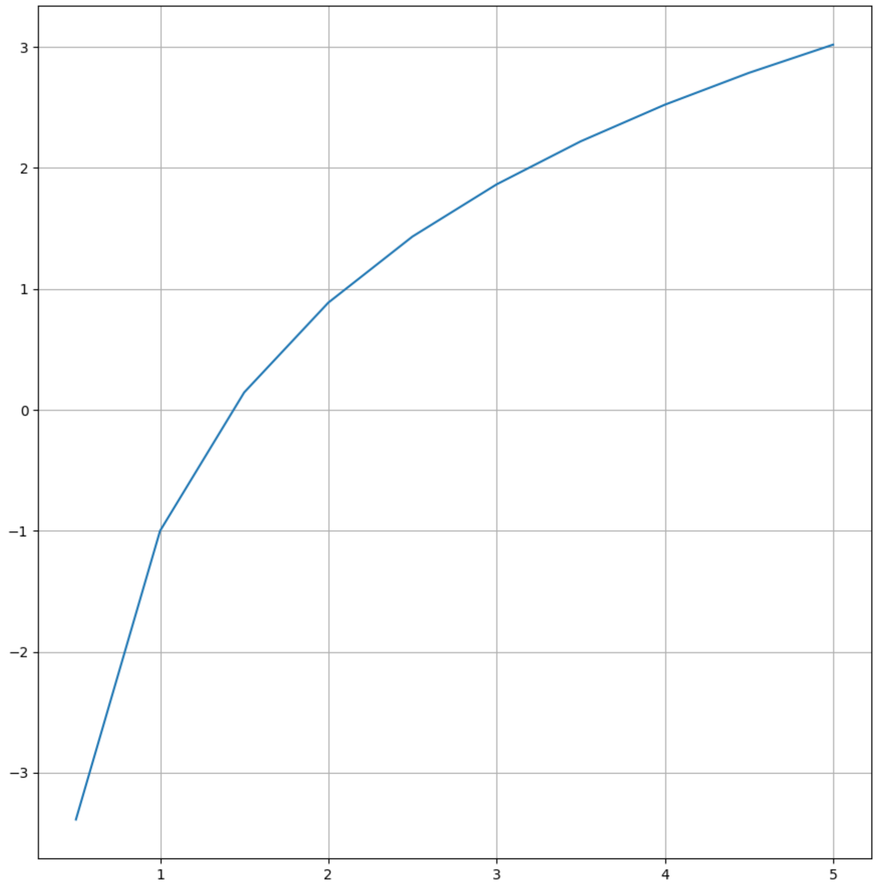



Таблица и график сделаны с помощью ЯП Python.

Построив график функции, определяем, что уравнение имеет только один корень, который находится в интервале $1 < x < 2$.

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

**<u>Метод Ньютона (метод касательных)</u>.**

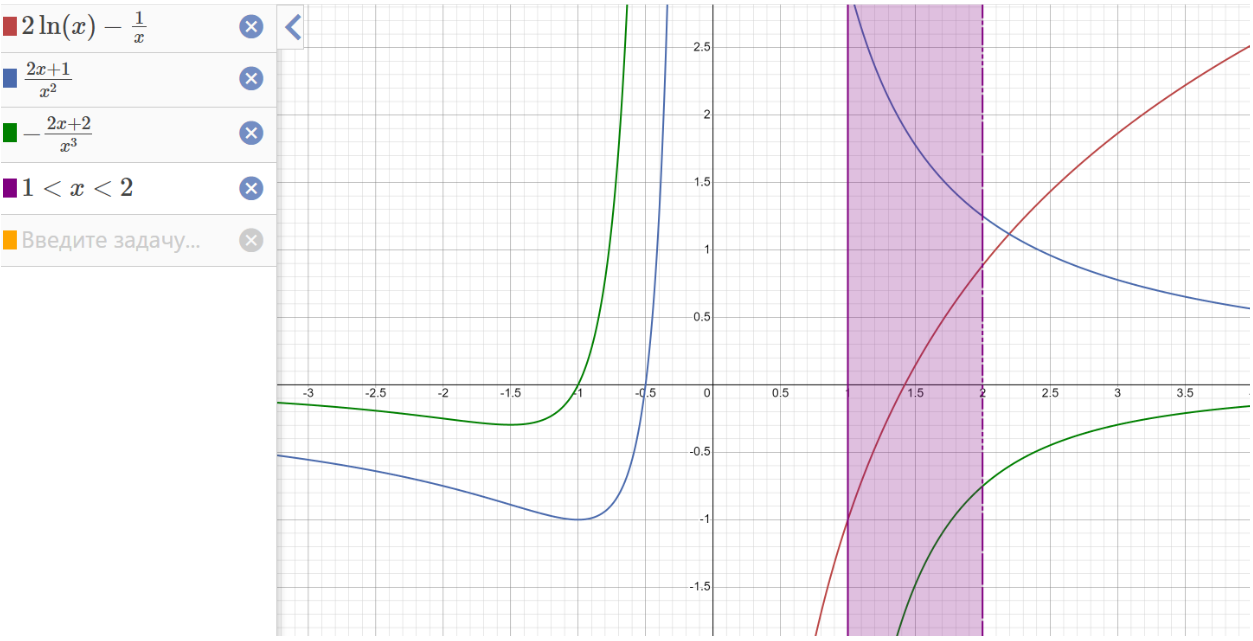
Определим поведение первой и второй производных функции $f(x)$ на интервале $[1, 2]$.

Для функции $f(x) = 2\ln x - \frac{1}{x}$

$$f'(x) = \frac{2x+1}{x^2}$$

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

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. $x_0 = 2$. Приближенные значения $x_k$ вычисляются по формуле:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}.$$

Итерации завершаются при выполнении условия $|x_{k+1} - x_k| < \varepsilon$. (в данном случае если взять левую границу – результат не изменится)

```
k    x(k)
0    2.0000000
1    1.2909645
2    1.4137211
3    1.4215029
4    1.4215299
5    1.4215299
Root: 1.4215299
```

**<u>Метод хорд.</u>** Вычисления проводятся по формуле $x_{k+1} = x_k - \frac{f(x_k)}{f(b) - f(x_k)}(b - x_k)$. Итерации завершаются при выполнении условия $|x_{k+1} - x_k| < \varepsilon$.

Начальное приближение $x_0 = a - \frac{(b-a)f(a)}{f(b)-f(a)}$.

```
k = 1, x(k) = 1.3616741
k = 2, x(k) = 1.4360907
k = 3, x(k) = 1.4180236
k = 4, x(k) = 1.4223763
k = 5, x(k) = 1.4213257
k = 6, x(k) = 1.4215792
k = 7, x(k) = 1.4215180
k = 8, x(k) = 1.4215328
k = 9, x(k) = 1.4215292
k = 10, x(k) = 1.4215301
k = 11, x(k) = 1.4215299
k = 12, x(k) = 1.4215299
Root is approximately 1.4215299
```

**Метод секущих.** В качестве начальных точек зададим: $x_0 = 1$ и $x_1 = 2$. Дальнейшие вычисления проводятся по формуле $x_{k+1} = x_k - \dfrac{f(x_k)}{f(x_k) - f(x_{k-1})}(x_k - x_{k-1})$. Итерации завершаются при выполнении условия $|x_{k+1} - x_k| < \varepsilon$.

```
k = 1, x(k) = 1.0000000
k = 2, x(k) = 2.0000000
k = 3, x(k) = 1.5301400
k = 4, x(k) = 1.3956934
k = 5, x(k) = 1.4227618
k = 6, x(k) = 1.4215441
k = 7, x(k) = 1.4215299
k = 8, x(k) = 1.4215299
Root: 1.4215299
```

**Конечноразностный метод Ньютона.** В качестве начального приближения берем $x_0 = 1$. Выбираем параметр $h = 0{,}05 > 0$. Вычисления проводятся по формуле $x_0 \in [a, b], \quad x_{k+1} = x_k - \dfrac{h \cdot f(x_k)}{f(x_k + h) - f(x_k)}, \quad h > 0 - \textit{малый параметр}$.

```
k = 1, x(k) = 1.3443541
k = 2, x(k) = 1.4205881
k = 3, x(k) = 1.4215503
k = 4, x(k) = 1.4215295
k = 5, x(k) = 1.4215299
k = 6, x(k) = 1.4215299
Root: 1.4215299
```

**<u>Метод Стеффенсена.</u>** В качестве начального приближения берем $x_0 = 2 \in [a,b]$

Вычисления проводятся по формуле $x_{k+1} = x_k - \dfrac{f^2(x_k)}{f\left(x_k + f(x_k)\right) - f(x_k)}$.

```
C:\Users\Vitaliy\AppData\Loc
k = 1, x(k) = 1.1145979
k = 2, x(k) = 1.2552522
k = 3, x(k) = 1.3804094
k = 4, x(k) = 1.4192832
k = 5, x(k) = 1.4215234
k = 6, x(k) = 1.4215299
k = 7, x(k) = 1.4215299
Root: 1.4215299
```

**<u>Метод простых итераций.</u>** Выбираем $x_0 = 1 \in [a,b]$. Вычисления проводятся по формуле $x_{k+1} = x_k - \tau f(x_k)$. Выбираем τ=0.5, удовлетворяющее условию $0 < \tau < \dfrac{2}{\min\left(f'(x)\right)}$.

```
k    x(k)
1    1.5000000
2    1.4278682
3    1.4218580
4    1.4215461
5    1.4215307
6    1.4215300
7    1.4215299
Root: 1.42152994
```

**Итоговая таблица**

| Метод решения | Выбранный интервал $[a, b]$ | Полученное решение | Количество итераций |
|---|---|---|---|
| 1. Метод Ньютона (метод касательных) | [1, 2] | 1.4215299 | 6 |
| 2. Метод хорд | [1, 2] | 1.4215299 | 12 |
| 3. Метод секущих | [1, 2] | 1.4215299 | 8 |
| 4. Конечноразностный метод Ньютона | [1, 2] | 1.4215299 | 6 |
| 5. Метод Стеффенсена | [1, 2] | 1.4215299 | 7 |
| 6. Метод простых итераций | [1, 2] | 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 $x \approx 1.4215299$

**ПРИЛОЖЕНИЕ**

***Программа построения таблицы значений функции***

```python
"""
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***

```python
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))
```

```python
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))
```

```python
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}')
```

```python
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}')
```

```python
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}')
```

```python
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}')
```