Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования

«Национальный исследовательский университет ИТМО»

Факультет Программной инженерии и компьютерной техники

Лабораторная работа №2

по дисциплине «Методы оптимизации»

Вариант: 3

Преподаватель: Селина Е. Г.

Выполнила: Вавилина Е. А.

Группа: P3230

Санкт-Петербург, 2025

Оглавление

Задание

Решить задачу четырьмя методами: методом половинного деления, методом золотого сечения, методом хорд и методом Ньютона. По 5 шагов каждого метода выполнить вручную + написать программу по каждому методу на одном из языков программирования.

Уравнение:

$$f(x) = \frac{1}{4}x^4 + x^2 - 8x + 12$$

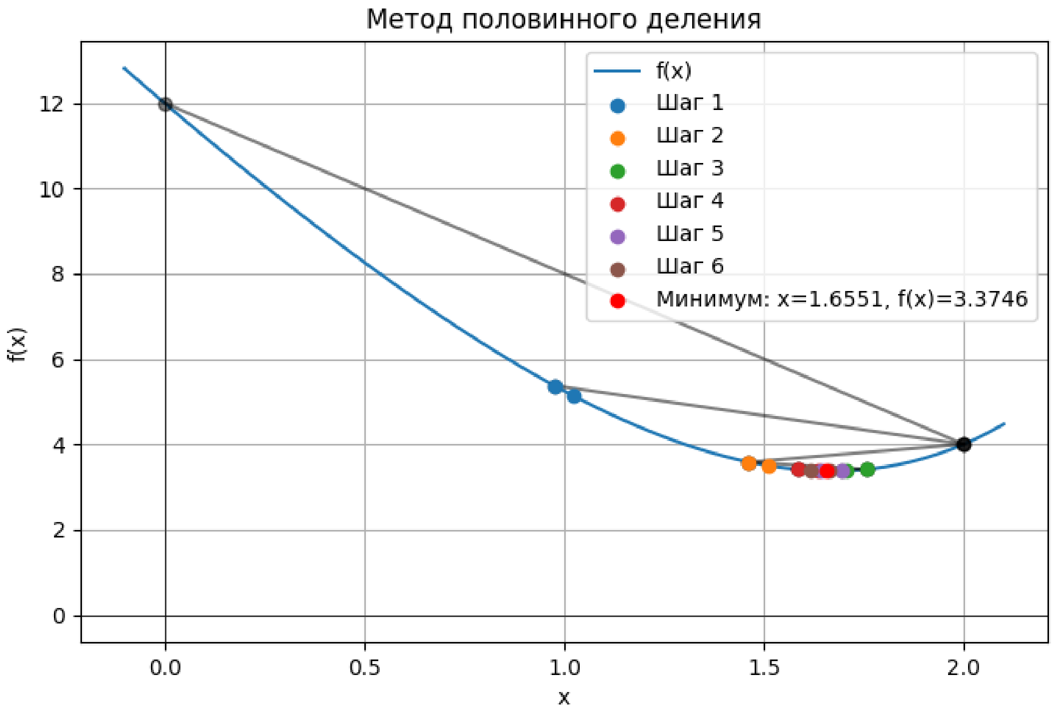
1. Метод половинного деления

Важное уточнение: расчет метода производился вручную, в процессе происходили округления при подсчетах значений. В результате этого получилось достичь нужной точности за 5 ходов, но пересчет через программу, которая обладает большей точностью, показал, что нужной длины можно достичь ТОЛЬКО к 7 шагу.

Ниже приведены расчеты без округлений:

	a	b									дельта к началу шага	
1	0,0000	2,0000	x_1 =	0,9750	f(x_1) =	5,3765	x_2 =	1,0250	f(x_2) =	5,1266	2,0000	
2	0,9750	2,0000	x_1 =	1,4625	f(x_1) =	3,5826	x_2 =	1,5125	f(x_2) =	3,4960	1,0250	
3	1,4625	2,0000	x_1 =	1,7063	f(x_1) =	3,3802	x_2 =	1,7563	f(x_2) =	3,4128	0,5375	
4	1,4625	1,7563	x_1 =	1,5844	f(x_1) =	3,4106	x_2 =	1,6344	f(x_2) =	3,3800	0,2938	
5	1,5844	1,7563	x_1 =	1,6453	f(x_1) =	3,3766	x_2 =	1,6953	f(x_2) =	3,3767	0,1719	
6	1,5844	1,6953	x_1 =	1,6148	f(x_1) =	3,3890	x_2 =	1,6648	f(x_2) =	3,3735	0,1109	не хватило точности
	1,6148	1,6953									0,0805	
	x_мин =	1,6551		f(x_мин) =	3,3746							

И график сдвигов, построенные программным методом:



Ниже приведены вычисление с погрешностью:

1) Метод половинного деления

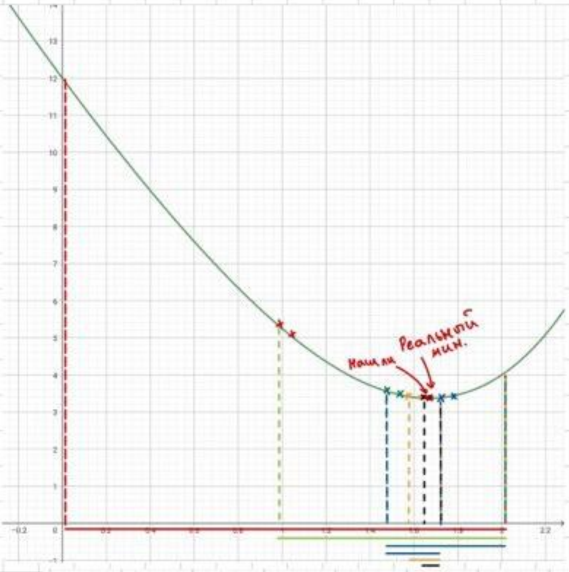
------- - стартовые границы

шаг 1: $x_1 = \frac{0+2-0,05}{2} = 0,975$ $x_2 = \frac{0+2+0,05}{2} = 1,025$

$f(x_1) \approx 5,377$ × > $f(x_2) \approx 5,127$ ×

Новый интервал: $[0,975; 2]$ -----

$2 - 0,975 > 2 \cdot \varepsilon \Rightarrow$ идем дальше

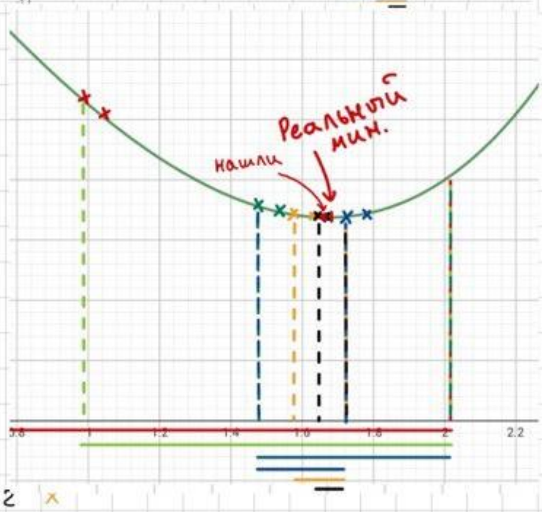


шаг 2: $x_1 = \frac{0,975+2-0,05}{2} = 1,4625$ $x_2 = 1,5125$

$f(x_1) = 3,583$ × > $f(x_2) = 3,496$ ×

Новый интервал: $[1,4625; 2]$ ------

$2 - 1,4625 > 2 \cdot \varepsilon \Rightarrow$ идем дальше



* шаг 3: $x_1 = \frac{1,4625+2-0,05}{2} = 1,706$ $x_2 = 1,765$

$f(x_1) = 3,38$ × < $f(x_2) = 3,413$ ×

Новый интервал: $[1,4625; 1,706]$ ------

$1,706 - 1,4625 > 2 \cdot \varepsilon \Rightarrow$ идем дальше

шаг 4: $x_1 = \frac{1,4625+1,706-0,05}{2} = 1,559$ $x_2 = 1,609$

$f(x_1) = 3,435$ × > $f(x_2) = 3,392$ ×

Новый интервал: $[1,559; 1,706]$ ------

$\underbrace{1,706 - 1,559}_{\approx 0,147} > 2 \cdot \varepsilon \Rightarrow$ идем дальше

* Начиная с п. 3 пошло накопление ошибки т.к. числа округлялись

Компьютерные вычисления показали, что для дост. нужной точности требуется 7 шагов

шаг 5: $x_1 = 1,6075$ $x_2 = 1,6575$

$f(x_1) = 3,39$ × > $f(x_2) = 3,374$ ×

Новый интервал: $[1,6075; 1,706]$ -----

$1,706 - 1,6075 = 0,0985 < 0,1 \Rightarrow$ закончили считать

Итого: $x_{min} = \frac{1,6075+1,706}{2} = 1,65675$ • $f(x_{min}) = 3,374$

Реальный минимум: $x = 1,67$ и $f(x) = 3,3734$

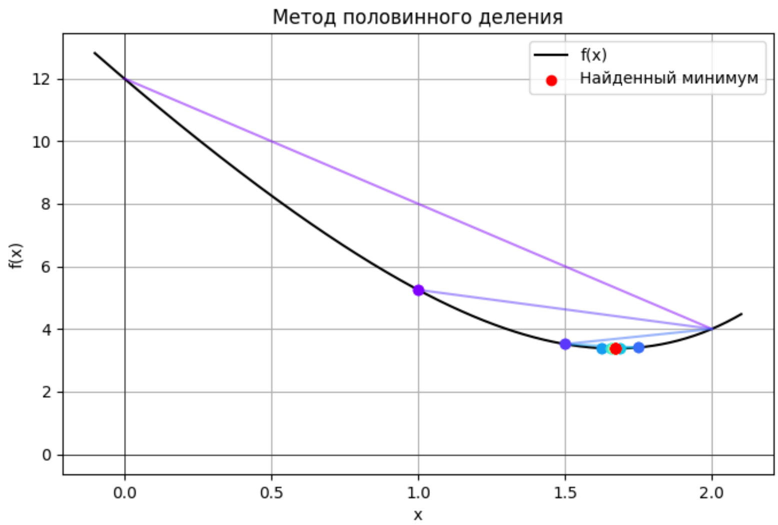
Реальный минимум найден исходя из построения и расчетов в программе geogebra.

Результаты работы программы (точка достижения минимума и значения в ней)

```
Метод половинного деления:
Точка минимума 1.670224056 ; Значение функции 3.3733904942985387

Метод половинного деления:

 Шаг        a        b       x1       x2       f1       f2
   1 0.000000 2.000000 0.999950 1.000050 5.250250 5.249750
   2 0.999950 2.000000 1.499925 1.500025 3.515747 3.515584
   3 1.499925 2.000000 1.749912 1.750012 3.407151 3.407237
   4 1.499925 1.750012 1.624919 1.625019 3.383887 3.383841
   5 1.624919 1.750012 1.687416 1.687516 3.374928 3.374946
   6 1.624919 1.687516 1.656167 1.656267 3.374413 3.374399
   7 1.656167 1.687516 1.671791 1.671891 3.373403 3.373405
   8 1.656167 1.671891 1.663979 1.664079 3.373594 3.373587
   9 1.663979 1.671891 1.667885 1.667985 3.373419 3.373417
  10 1.667885 1.671891 1.669838 1.669938 3.373391 3.373391
  11 1.669838 1.671891 1.670815 1.670915 3.373392 3.373393
  12 1.669838 1.670915 1.670327 1.670427 3.373391 3.373391
  13 1.669838 1.670427 1.670083 1.670183 3.373391 3.373391
  14 1.670083 1.670427 1.670205 1.670305 3.373391 3.373391
  15 1.670083 1.670305 1.670144 1.670244 3.373391 3.373390

==================================================
```

График, содержащий все точки, нанесенные на график функции. Точки, полученные на одном шаге окрашены одним цветом. Но они неразличимы в силу малой погрешности, поэтому сливаются в одну.

Линии стягивают области графика, попавшие в интервал на каждом шаге.



2. Метод золотого сечения

Ниже приведены расчеты без округлений:

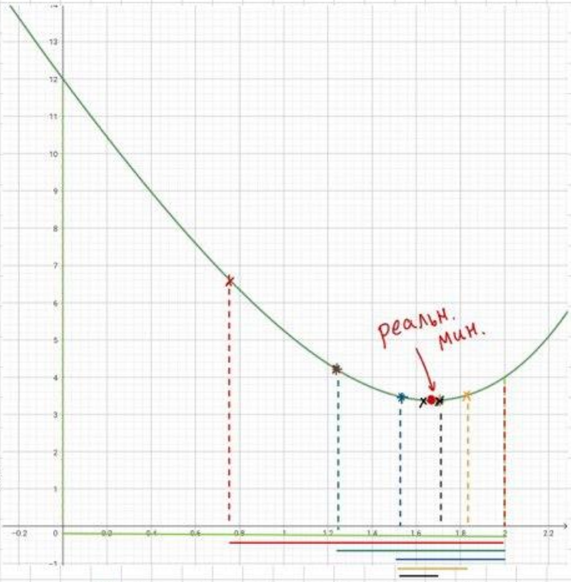
	a	b										
			фи =	1,618034		1/(фи + 1) =	0,382					
											дельта к началу шага	
1	0,0000	2,0000	x_1 =	0,7639	f(x_1) =	6,5573	x_2 =	1,2361	f(x_2) =	4,2229	2,0000	
2	0,7639	2,0000	x_1 =	1,2361	f(x_1) =	4,2229	x_2 =	1,5279	f(x_2) =	3,4738	1,2361	
3	1,2361	2,0000	x_1 =	1,5279	f(x_1) =	3,4738	x_2 =	1,7082	f(x_2) =	3,3810	0,7639	
4	1,5279	2,0000	x_1 =	1,7082	f(x_1) =	3,3810	x_2 =	1,8197	f(x_2) =	3,4948	0,4721	
5	1,5279	1,8197	x_1 =	1,6393	f(x_1) =	3,3783	x_2 =	1,7082	f(x_2) =	3,3810	0,2918	
6	1,5279	1,7082	x_1 =	1,5967	f(x_1) =	3,4007	x_2 =	1,6393	f(x_2) =	3,3783	0,1803	не хватило точности
7	1,5967	1,7082	x_1 =	1,6393	f(x_1) =	3,3783	x_2 =	1,6656	f(x_2) =	3,3735	0,1115	
8	1,6393	1,7082	x_1 =	1,6656	f(x_1) =	3,3735	x_2 =	1,6819	f(x_2) =	3,3741	0,0689	
	1,6393	1,6819									0,0426	
	x_мин =	1,6606		f(x_мин) =	3,3739							

Нанесение вычислений на график:

2) Метод золотого сечения

$\varphi = \frac{1+\sqrt{5}}{2}$ $\quad \frac{1}{\varphi+1} \approx 0{,}382$

$x_1 = a + \frac{b-a}{\varphi+1}$ $\quad x_2 = b - \frac{b-a}{\varphi+1}$



шаг 1: $x_1 = 0{,}7639$ $\quad x_2 = 1{,}2361$

$f(x_1) = 6{,}5573 > f(x_2) = 4{,}2229$

Новый интервал: $[0{,}7639;\ 2]$

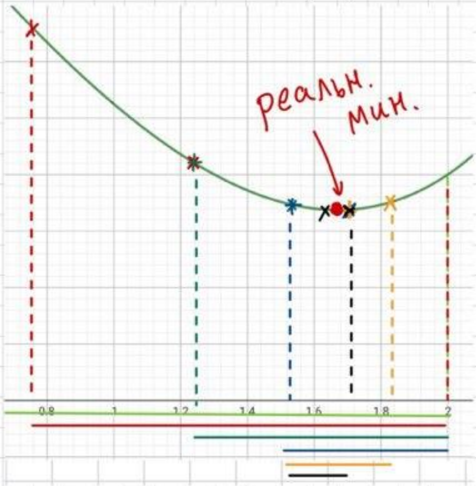
$2 - 0{,}7639 > 0{,}05$

шаг 2: $x_1 = 1{,}2361$ $\quad x_2 = 1{,}5279$

$f(x_1) = 4{,}2229 > f(x_2) = 3{,}4738$

Новый интервал: $[1{,}2361;\ 2]$

$2 - 1{,}2361 > 0{,}05$



шаг 3: $x_1 = 1{,}5279$ $\quad x_2 = 1{,}7082$

$f(x_1) = 3{,}4738 > f(x_2) = 3{,}3810$

Новый интервал: $[1{,}5279;\ 2]$

$2 - 1{,}5279 > 0{,}05$

шаг 4: $x_1 = 1{,}7082$ $\quad x_2 = 1{,}8197$

$f(x_1) = 3{,}3810 < f(x_2) = 3{,}4948$

Новый интервал: $[1{,}5279;\ 1{,}8197]$

$1{,}8197 - 1{,}5279 > 0{,}05$

шаг 5: $x_1 = 1{,}6393$ $\quad x_2 = 1{,}7082$

$f(x_1) = 3{,}3783 < f(x_2) = 3{,}3810$

Новый интервал: $[1{,}5279;\ 1{,}7082]$

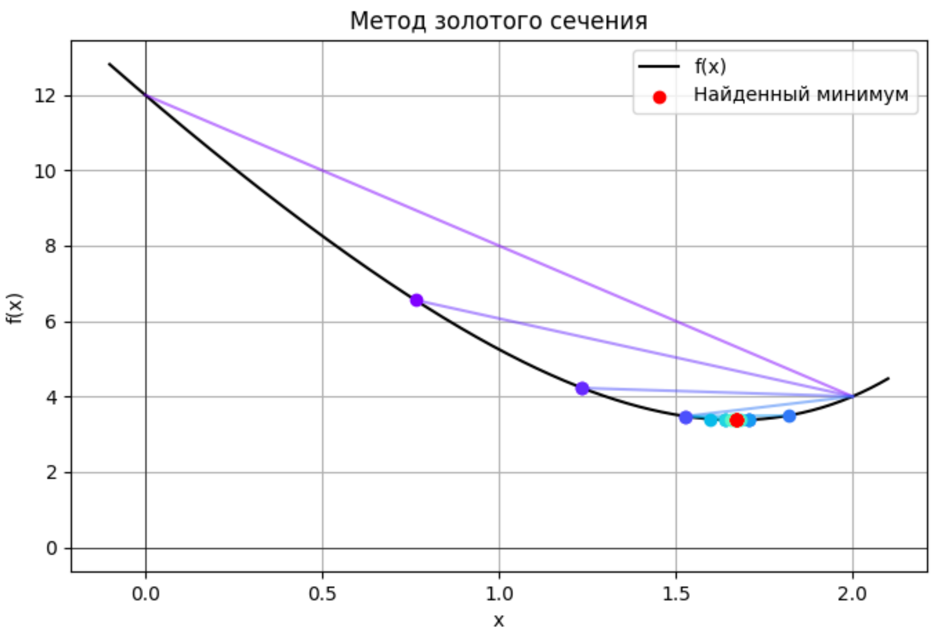
$1{,}7082 - 1{,}5279 > 0{,}05$

Точность не достигнута

Результаты работы программы (точка достижения минимума и значения в ней)

```
Точка минимума 1.670228605 ; Значение функции 3.373390493432213

Метод золотого сечения:

 Шаг        a        b       x1       x2       f1       f2
   1 0.000000 2.000000 0.763932 1.236068 6.557281 4.222912
   2 0.763932 2.000000 1.236068 1.527864 4.222912 3.473775
   3 1.236068 2.000000 1.527864 1.708204 3.473775 3.380953
   4 1.527864 2.000000 1.708204 1.819660 3.380953 3.494832
   5 1.527864 1.819660 1.639320 1.708204 3.378299 3.380953
   6 1.527864 1.708204 1.596748 1.639320 3.400741 3.378299
   7 1.596748 1.708204 1.639320 1.665631 3.378299 3.373501
   8 1.639320 1.708204 1.665631 1.681893 3.373501 3.374097
   9 1.639320 1.681893 1.655581 1.665631 3.374500 3.373501
  10 1.655581 1.681893 1.665631 1.671843 3.373501 3.373404
  11 1.665631 1.681893 1.671843 1.675681 3.373404 3.373544
  12 1.665631 1.675681 1.669470 1.671843 3.373394 3.373404
  13 1.665631 1.671843 1.668004 1.669470 3.373417 3.373394
  14 1.668004 1.671843 1.669470 1.670376 3.373394 3.373391
  15 1.669470 1.671843 1.670376 1.670936 3.373391 3.373393
  16 1.669470 1.670936 1.670030 1.670376 3.373391 3.373391
  17 1.670030 1.670936 1.670376 1.670590 3.373391 3.373391
  18 1.670030 1.670590 1.670244 1.670376 3.373390 3.373391
  19 1.670030 1.670376 1.670162 1.670244 3.373391 3.373390
  20 1.670162 1.670376 1.670244 1.670295 3.373390 3.373391
```

График, содержащий все точки, нанесенные на график функции. Точки, полученные на одном шаге окрашены одним цветом. Но они неразличимы в силу малой погрешности, поэтому сливаются в одну.
Линии стягивают области графика, попавшие в интервал на каждом шаге.



3. Метод хорд

Ниже приведены расчеты без округлений:

	a	b									дельта в конце шага	
1	0,0000	2,0000	x =	1,3333	f'(x) =	-2,9630	f'(a) =	-8,0000	f'(b) =	4,0000	2,9630	
2	1,3333	2,0000	x =	1,6170	f'(x) =	-0,5378	f'(a) =	-2,9630	f'(b) =	4,0000	0,5378	
3	1,6170	2,0000	x =	1,6624	f'(x) =	-0,0809	f'(a) =	-0,5378	f'(b) =	4,0000	0,0809	
4	1,6624	2,0000	x =	1,6691	f'(x) =	-0,0118	f'(a) =	-0,0809	f'(b) =	4,0000	0,0118	Закончили
	x_мин =	1,6691		f(x_мин) =	3,3734							

Нанесение вычислений на график:

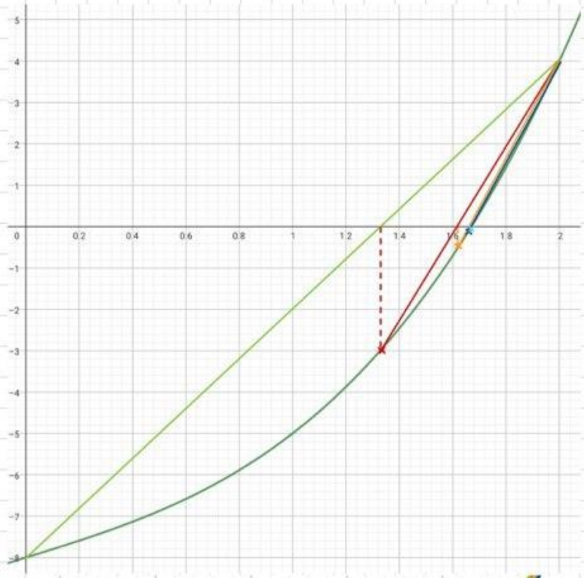
3) Метод хорд

Уравнение прямой через 2 точки $y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$

$\Rightarrow y - f'(a) = \frac{f'(b) - f'(a)}{b - a}(x - a)$; $y = 0$ и $x = \tilde{x}$

$\Rightarrow \tilde{x} = a - \frac{f'(a)}{f'(b) - f'(a)}(b - a)$

Производная функции: $f'(x) = x^3 + 2x - 8$



шаг 1: $f'(a) = -8$ $f'(b) = 4$

$x = 1{,}3333$, $f'(x) = -2{,}963 < 0$

$|-2{,}963| > 0{,}05$, продолжаем

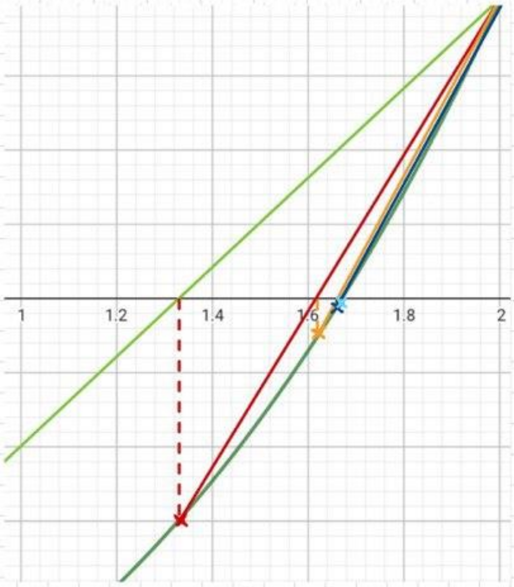
новый интервал: $[1{,}3333; 2]$

шаг 2: $f'(a) = -2{,}963$ $f'(b) = 4$

$x = 1{,}617$, $f'(x) = -0{,}5378 < 0$

$|-0{,}5378| > 0{,}05$

новый интервал: $[1{,}617; 2]$



шаг 3: $f'(a) = -0{,}5378$ $f'(b) = 4$

$x = 1{,}6624$, $f'(x) = -0{,}0809$

$|-0{,}0809| > 0{,}05$

новый интервал: $[1{,}6624; 2]$

шаг 4: $f'(a) = -0{,}0809$ $f'(b) = 4$

$x = 1{,}6691$, $f'(x) = -0{,}0118$

$|-0{,}0118| < 0{,}05$

$\Rightarrow$ точность достигнута

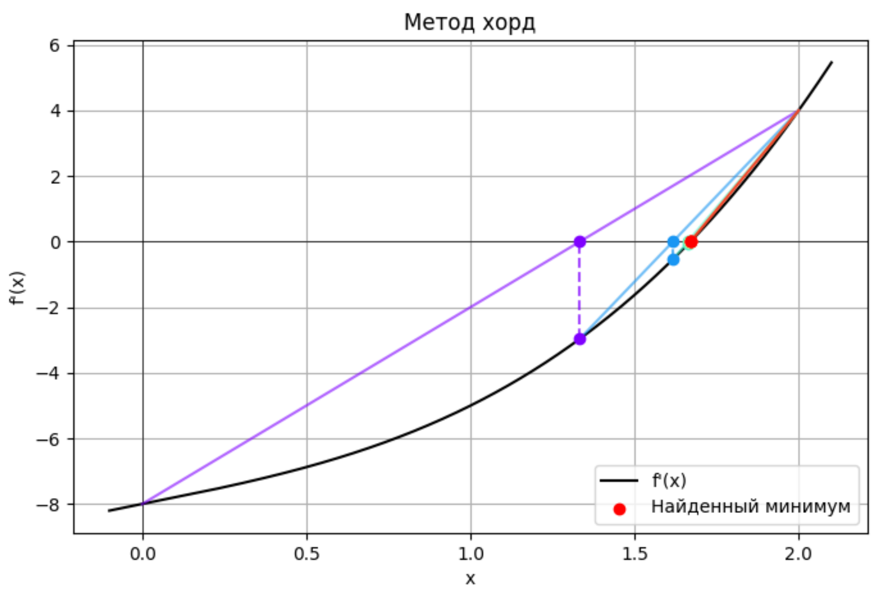
$\Rightarrow$ минимум в $x_{min} = 1{,}6691$

и равен $3{,}3734$

Результаты работы программы (точка достижения минимума и значения в ней)

```
Метод хорд:
Точка минимума 1.67024121 ; Значение функции 3.3733904921527067

Метод хорд:

 Шаг        a   b         x       f(x)
   1 0.000000 2.0 1.333333 -2.962963
   2 1.333333 2.0 1.617021 -0.537838
   3 1.617021 2.0 1.662413 -0.080900
   4 1.662413 2.0 1.669105 -0.011806
   5 1.669105 2.0 1.670079 -0.001715
   6 1.670079 2.0 1.670221 -0.000249

==================================================
```

График, содержащий все точки, нанесенные на график функции.
Линии стягивают области графика, попавшие в интервал на каждом шаге.



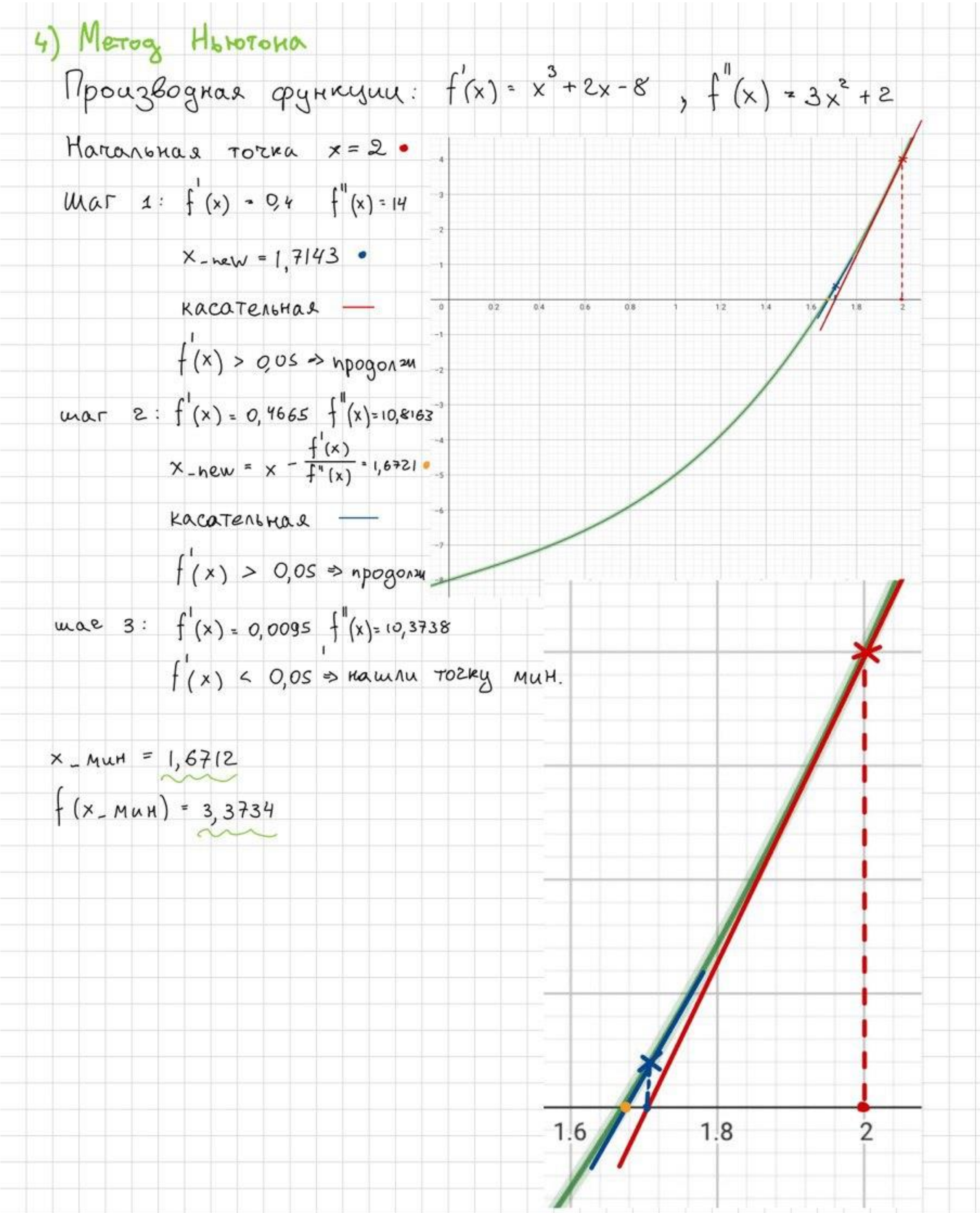
4. Метод Ньютона

Ниже приведены расчеты без округлений. В зависимости от точки первого приближения возможно получение разных результатов за разное число шагов.

							дельта в конце шага	
1	x =	0,0000	f'(x) =	-8,0000	f''(a) =	2,0000	8,0000	
2	x =	4,0000	f'(x) =	64,0000	f''(a) =	50,0000	64,0000	
3	x =	2,7200	f'(x) =	17,5636	f''(a) =	24,1952	17,5636	
4	x =	1,9941	f'(x) =	3,9174	f''(a) =	13,9291	3,9174	
5	x =	1,7128	f'(x) =	0,4509	f''(a) =	10,8015	0,4509	Не хватило точности
6	x =	1,6711	f'(x) =	0,0089	f''(a) =	10,3777	0,0089	Закончили
	x_мин =	1,6711		f(x_мин) =	3,3734			

							дельта в конце шага	
1	x =	2,0000	f'(x) =	4,0000	f''(a) =	14,0000	4,0000	
2	x =	1,7143	f'(x) =	0,4665	f''(a) =	10,8163	0,4665	
3	x =	1,6712	f'(x) =	0,0095	f''(a) =	10,3783	0,0095	
	x_мин =	1,6712		f(x_мин) =	3,3734			

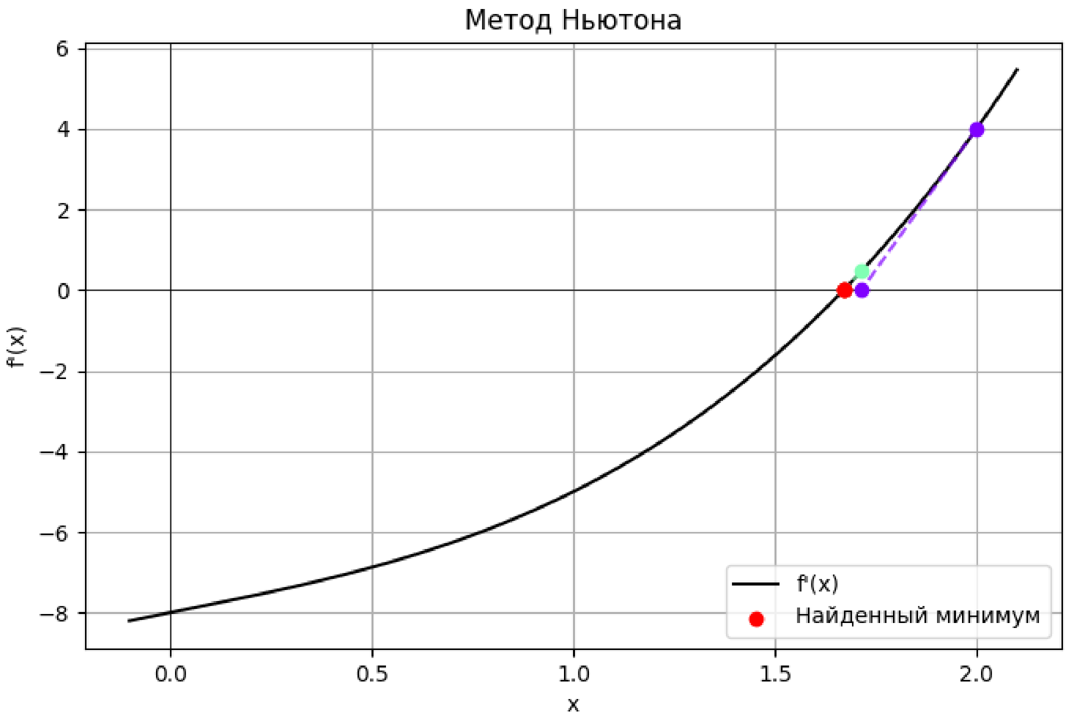
Нанесение вычислений на график (для 2 способа):



Результаты работы программы (точка достижения минимума и значения в ней)

```
Метод Ньютона:
Точка минимума 1.670245101 ; Значение функции 3.3733904920905147

Метод Ньютона:

 Шаг  x_старый  x_новый    f'(x)    f''(x)
   1  2.000000 1.714286 4.000000 14.000000
   2  1.714286 1.671159 0.466472 10.816327
   3  1.671159 1.670245 0.009485 10.378318

==================================================
```

График, содержащий все точки, нанесенные на график функции и касательные к функции в этих точках для поиска следующей.



5. Итоговый код

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from decimal import Decimal, getcontext

# Устанавливаем точность для Decimal
getcontext().prec = 10

def f(x):
    return (1 / 4) * x ** 4 + x ** 2 - 8 * x + 12

def f_pr(x):
    return x ** 3 + 2 * x - 8
```

```python
def f_pr2(x):
    return 3 * x ** 2 + 2

# Половинное
def refined_bisection_method(a, b, eps):
    iterations = []

    a, b, eps = Decimal(a), Decimal(b), Decimal(eps)

    while (b - a) > 2 * eps:
        x1 = (a + b - eps) / 2
        x2 = (a + b + eps) / 2

        f1, f2 = f(float(x1)), f(float(x2))  # Преобразуем обратно в float
для вычисления функции
        iterations.append((float(a), float(b), float(x1), float(x2), f1, f2))

        if f1 > f2:
            a = x1
        else:
            b = x2

    # Приближённый минимум
    x_min = (a + b) / 2
    f_min = f(float(x_min))

    return float(x_min), float(f_min), iterations

# Золотое
def golden_section_method(a, b, eps):
    iterations = []
    phi = Decimal((1 + np.sqrt(5)) / 2)

    a, b, eps = Decimal(a), Decimal(b), Decimal(eps)

    x1 = b - (b - a) / phi
    x2 = a + (b - a) / phi
    f1, f2 = f(float(x1)), f(float(x2))

    while (b - a) > 2 * eps:
        iterations.append((float(a), float(b), float(x1), float(x2), f1, f2))

        if f1 < f2:
            b, x2, f2 = x2, x1, f1
            x1 = b - (b - a) / phi
            f1 = f(float(x1))
        else:
            a, x1, f1 = x1, x2, f2
            x2 = a + (b - a) / phi
            f2 = f(float(x2))

    # Приближённый минимум
    x_min = (a + b) / 2
    f_min = f(float(x_min))
    return float(x_min), float(f_min), iterations


# Хорды
def secant_method(a, b, eps):
    iterations = []
    a, b, eps = Decimal(a), Decimal(b), Decimal(eps)

    while True:
        x_new = a - (Decimal(f_pr(float(a))) * (a - b)) /
```

```
(Decimal(f_pr(float(a))) - Decimal(f_pr(float(b))))
        f_new = f_pr(float(x_new))

        if abs(f_new) < eps:
            break

        iterations.append((float(a), float(b), float(x_new), f_new))

        if f_new > 0:
            b = x_new
        else:
            a = x_new

    # Приближённый минимум
    x_min = float(x_new)
    f_min = f(float(x_min))

    return x_min, f_min, iterations

# Ньютон
def newton_method(x0, eps):
    iterations = []
    x = Decimal(x0)
    eps = Decimal(eps)

    while True:
        f1 = Decimal(f_pr(float(x)))
        f2 = Decimal(f_pr2(float(x)))

        if abs(f1) < eps:
            break

        x_new = x - f1 / f2  # Шаг метода Ньютона
        iterations.append((float(x), float(x_new), float(f1), float(f2)))

        x = x_new

    x_min = float(x)
    f_min = f(x_min)

    return x_min, f_min, iterations

# График средних и золотого
def print_plt(x, fnk, steps, method_name):
    x_vals = np.linspace(a - 0.1, b + 0.1, 400)
    y_vals = f(x_vals)

    plt.figure(figsize=(8, 5))
    plt.plot(x_vals, y_vals, label="f(x)", color="black")
    plt.axhline(0, color="black", linewidth=0.5)
    plt.axvline(0, color="black", linewidth=0.5)

    colors = plt.cm.rainbow(np.linspace(0, 1, len(steps)))

    for i, (a_i, b_i, x1_i, x2_i, f1_i, f2_i) in enumerate(steps):
        plt.scatter([x1_i, x2_i], [f1_i, f2_i], color=colors[i], zorder=3)
        plt.plot([a_i, b_i], [f(a_i), f(b_i)], color=colors[i], alpha=0.5)

    plt.scatter([x], [fnk], color="red", label=f"Найденный минимум",
zorder=3)
    plt.xlabel("x")
    plt.ylabel("f(x)")
    plt.title(f"{method_name}")
    plt.legend()
```

```
    plt.grid()
    plt.show()

# График хорды
def print_plt_secant(x, fnk, steps, method_name):
    x_vals = np.linspace(a - 0.1, b + 0.1, 400)
    y_vals = f_pr(x_vals)

    plt.figure(figsize=(8, 5))
    plt.plot(x_vals, y_vals, label="f'(x)", color="black")
    plt.axhline(0, color="black", linewidth=0.5)
    plt.axvline(0, color="black", linewidth=0.5)

    colors = plt.cm.rainbow(np.linspace(0, 1, len(steps)))

    for i, (a_i, b_i, x_new, f_new) in enumerate(steps):
        plt.scatter([x_new, x_new], [f_new, 0], color=colors[i], zorder=3)
        plt.plot([x_new, x_new], [f_new, 0], linestyle="dashed",
color=colors[i], alpha=0.7)
        plt.plot([a_i, b_i], [f_pr(a_i), f_pr(b_i)],  color=colors[i],
alpha=0.6)

    plt.scatter([x], 0, color="red", label=f"Найденный минимум", zorder=3)
    plt.xlabel("x")
    plt.ylabel("f'(x)")
    plt.title(f"{method_name}")
    plt.legend()
    plt.grid()
    plt.show()

# Строим Ньютона
def print_plt_newton(x, fnk, steps, method_name):
    x_vals = np.linspace(a - 0.1, b + 0.1, 400)
    y_vals = f_pr(x_vals)

    plt.figure(figsize=(8, 5))
    plt.plot(x_vals, y_vals, label="f'(x)", color="black")
    plt.axhline(0, color="black", linewidth=0.5)
    plt.axvline(0, color="black", linewidth=0.5)

    colors = plt.cm.rainbow(np.linspace(0, 1, len(steps)))

    for i, (x_old, x_new, f1, f2) in enumerate(steps):
        plt.scatter([x_new], [0], color=colors[i], zorder=3)
        plt.scatter([x_old], [f_pr(x_old)], color=colors[i], zorder=3)
        plt.plot([x_old, x_new], [f_pr(x_old), 0], linestyle="dashed",
color=colors[i], alpha=0.7)

    plt.scatter([x], 0, color="red", label=f"Найденный минимум", zorder=3)
    plt.xlabel("x")
    plt.ylabel("f'(x)")
    plt.title(f"{method_name}")
    plt.legend()
    plt.grid()
    plt.show()

# Красивые выводы
def print_iterations(steps, method_name):
    if not steps:
        print(f"\n{method_name} - нет итераций (метод сошелся сразу).\n")
        return

    col_count = len(steps[0])
```

```
    if method_name == "Метод Ньютона" and col_count == 4:
        columns = ["Шаг", "x_старый", "x_новый", "f'(x)", "f''(x)"]
    elif method_name == "Метод хорд" and col_count == 4:
        columns = ["Шаг", "a", "b", "x", "f(x)"]
    elif method_name == "Метод золотого сечения" and col_count == 6:
        columns = ["Шаг", "a", "b", "x1", "x2", "f1", "f2"]
    elif method_name == "Метод половинного деления" and col_count == 6:
        columns = ["Шаг", "a", "b", "x1", "x2", "f1", "f2"]
    else:
        print(f"Ошибка: несоответствие количества данных и названий столбцов
в {method_name}")
        return

    steps_with_index = [(i + 1,) + step for i, step in enumerate(steps)]
    df = pd.DataFrame(steps_with_index, columns=["Шаг"] + columns[1:])

    print(f"\n{method_name}:\n")
    print(df.to_string(index=False))
    print("\n" + "=" * 50 + "\n")


# Начальные параметры
a, b = 0, 2
eps = 0.0001

# Половинное деление
x_min, f_min, refined_bisection_steps = refined_bisection_method(a, b, eps)
print("Метод половинного деления:")
print("Точка минимума", x_min, "; Значение функции", f_min)
print_iterations(refined_bisection_steps, "Метод половинного деления")
print_plt(x_min, f_min, refined_bisection_steps, "Метод половинного деления")

# Метод золотого сечения
x_golden, f_golden, golden_section_steps = golden_section_method(a, b, eps)
print("Метод золотого сечения:")
print("Точка минимума", x_golden, "; Значение функции", f_golden)
print_iterations(golden_section_steps, "Метод золотого сечения")
print_plt(x_golden, f_golden, golden_section_steps, "Метод золотого сечения")

# Метода хорд
x_secant, f_secant, secant_steps = secant_method(a, b, eps)
print("Метод хорд:")
print("Точка минимума", x_secant, "; Значение функции", f_secant)
print_iterations(secant_steps, "Метод хорд")
print_plt_secant(x_secant, f_secant, secant_steps, "Метод хорд")

# Метод Ньютона
x_newton, f_newton, newton_steps = newton_method(2, eps)
print("Метод Ньютона:")
print("Точка минимума", x_newton, "; Значение функции", f_newton)
print_iterations(newton_steps, "Метод Ньютона")
print_plt_newton(x_newton, f_newton, newton_steps, "Метод Ньютона")
```

Вывод

1. При ручном подсчете единственный метод, который позволил гарантированно получить значение с нужной точностью за 5 ходов – метод хорд. Метод Ньютона сильно зависит от выбора стартовой точки.
2. Наиболее точный метод после компьютерных вычислений – Метод Ньютона со стартовой точкой 2.