МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра информационных технологий**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1**

**по дисциплине  
 «Методы оптимизации»**

Выполнил студент группы 35/2                                       Д.А. Вербицкий

Направление подготовки  02.03.03  Математическое обеспечение и администрирование информационных систем

Курс    3

Краснодар

2025 г.

# 2 вариант

# Постановка задачи

Найти минимум функции:

f(x) = x2-2\*x+5

на заданном отрезке [-2;8], используя методы дихотомии, Фибоначчи и золотого сечения.

График функции имеет следующий вид:

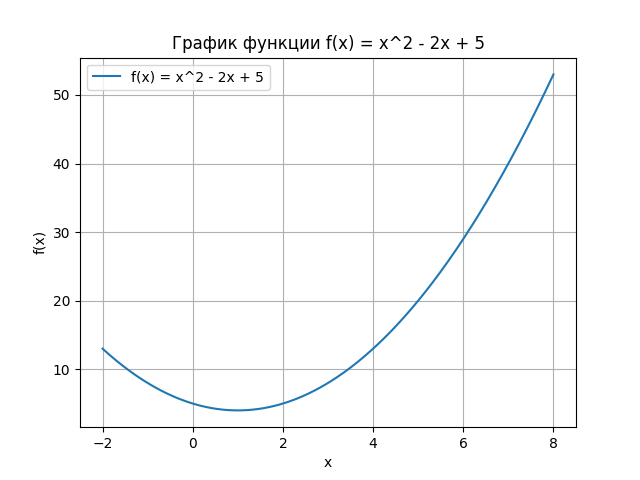


Рисунок 1 – график функции f(x)

# Метод Дихотомии

**Шаг 1.** Задать начальный интервал неопределённости L0=[a0,b0], при этом ε>0 (малое число), l>0 (требуемая точность).

**Шаг 2.** Положить k=0.

**Шаг 3.** Вычислить

yk= (ak+bk−ε) / 2, f(yk), zk=(ak+bk+ε)/2, f(zk).

**Шаг 4.** Сравнить f(yk) с f(zk):

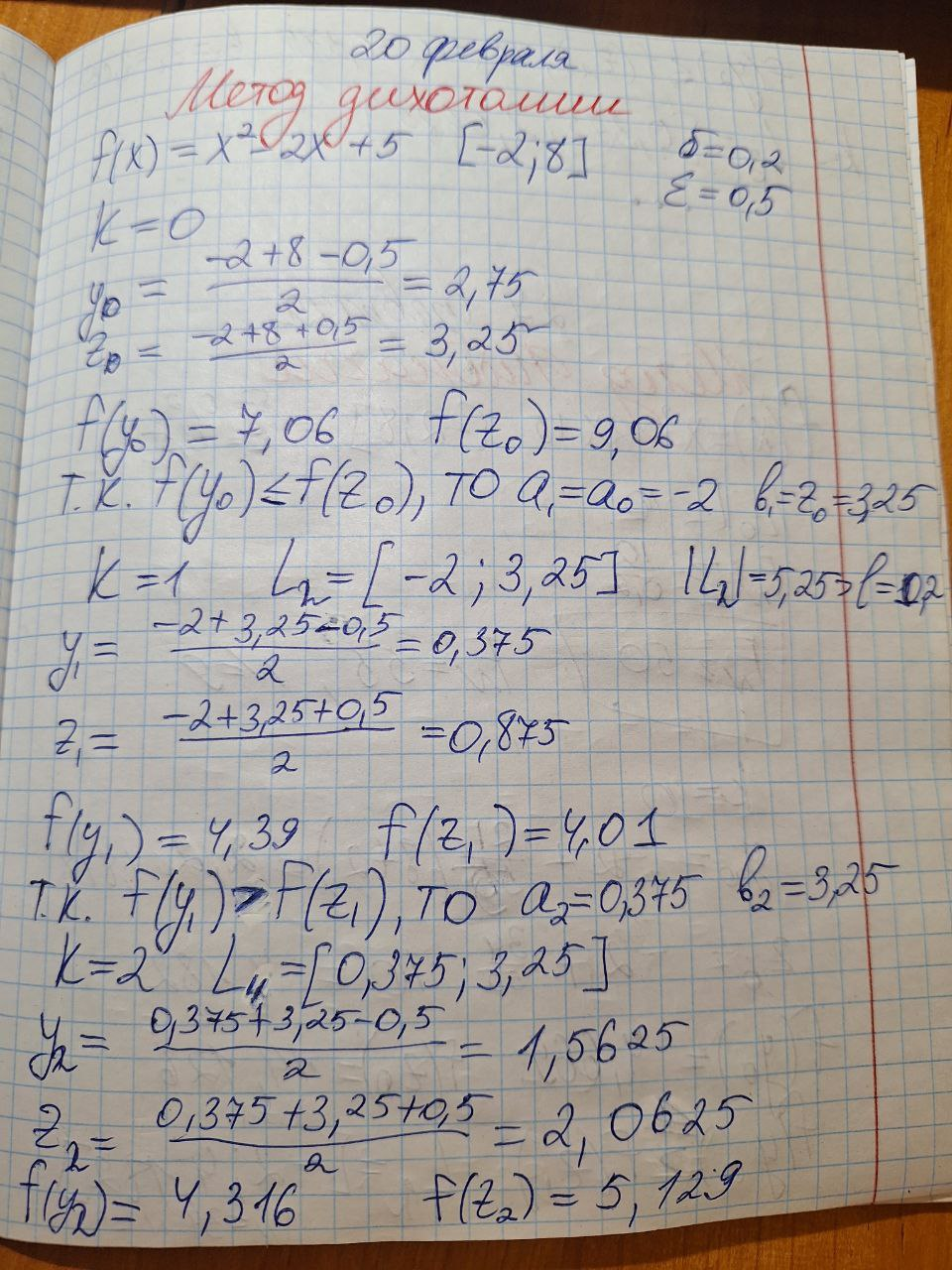
1. Если f(yk) ≤ f(zk), положить ak+1=ak, bk+1=zk и перейти к шагу 5.
2. Если f(yk) > f(zk), положить ak+1=yk,bk+1=bk и также перейти к шагу 5.

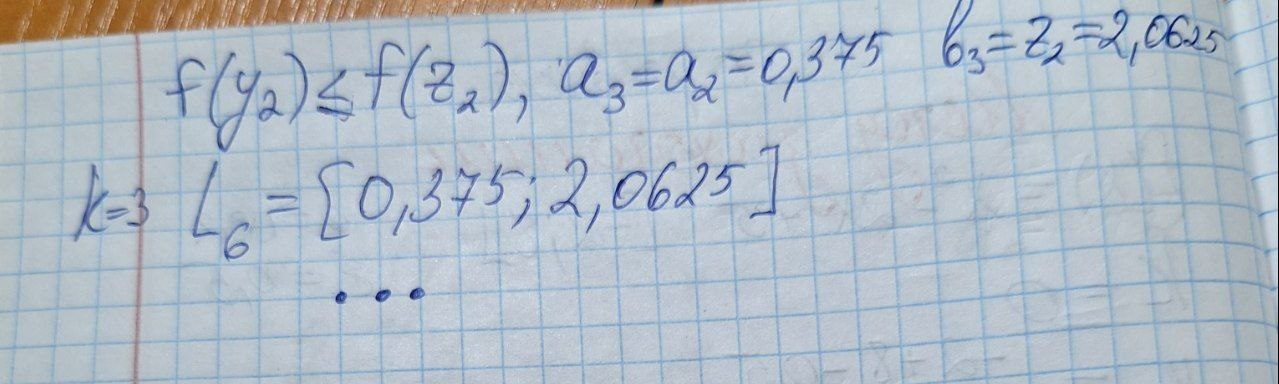
**Шаг 5.** Вычислить длину нового интервала:

∣L2(k+1)∣=∣bk+1−ak+1∣

Далее проверить условие окончания:

1. Если ∣L2(k+1)∣≤l, процесс поиска завершается и x∗∈L2(k+1)=[ ak+1, bk+1]. В качестве приближённого решения можно взять середину последнего интервала: x∗=(ak+1+bk+1)/ 2
2. Если ∣L2(k+1)∣>l, положить k=k+1 и вернуться к шагу 3.





## Программный код

N = 0  
k = -1  
  
  
def f(arg):  
 global N  
 N = 2 \* (k + 1)  
 return arg \* arg - 2 \* arg + 5  
  
  
epsilon = 0.1  
l = 0.2  
a = -2  
b = 8  
  
while b - a > l:  
 k += 1  
 y = (a + b - epsilon) / 2  
 z = (a + b + epsilon) / 2  
 if f(y) <= f(z):  
 b = z  
 else:  
 a = y  
  
x = (a + b) / 2  
print(f"Минимум находится в точке {x} и равен {f(x)}")  
print(f"k = {k}")  
print(f"N = {N}")  
print(f"R(N) = {1 / (2 \*\* (N / 2))}")  
print(a, ' ', b)

## Вывод на консоль:



# Метод Фибоначчи

**Шаг 1.** Задать начальный интервал неопределенности:

L0 = [a0, b0]

l>0 — допустимая длина конечного интервала;

ε>0 — константа различимости.

**Шаг 2.** Определить количество вычислений функции N:

Найти наименьшее целое N, удовлетворяющее условию: FN≥∣L0∣/l​, где FN​ — число Фибоначчи;

Сгенерировать последовательность чисел Фибоначчи F0, F1,...,FN

**Шаг 3.** Инициализировать счетчик: k=0.

**Шаг 4.** Вычислить начальные точки:

y0=a0+FN−2 / FN \* (b0−a0); z0= a0+FN−1 / FN \* (b0−a0).

**Шаг 5.** Вычислить значения функции в точках yk​ и zk​: f(yk), f(zk).

**Шаг 6.** Сравнить f(yk) и f(zk):

Если f(yk)≤f(zk):

ak+1=ak

bk+1=zk

zk+1=yk

yk+1=ak+1+FN−k−2/FN−k−1(bk+1−ak+1)

Перейти к **шагу 7**.

Если f(yk)>f(zk):

ak+1=yk

bk+1=bk

yk+1=zk

zk+1=ak+1+FN−k−2/FN−k−1(bk+1−ak+1)

Перейти к **шагу 7**.

**Шаг 7.** Проверить условие завершения:

* **Если k≠N−3:**
  + k=k+1;
  + Вернуться к **шагу 5**.
* **Если k=N−3:**

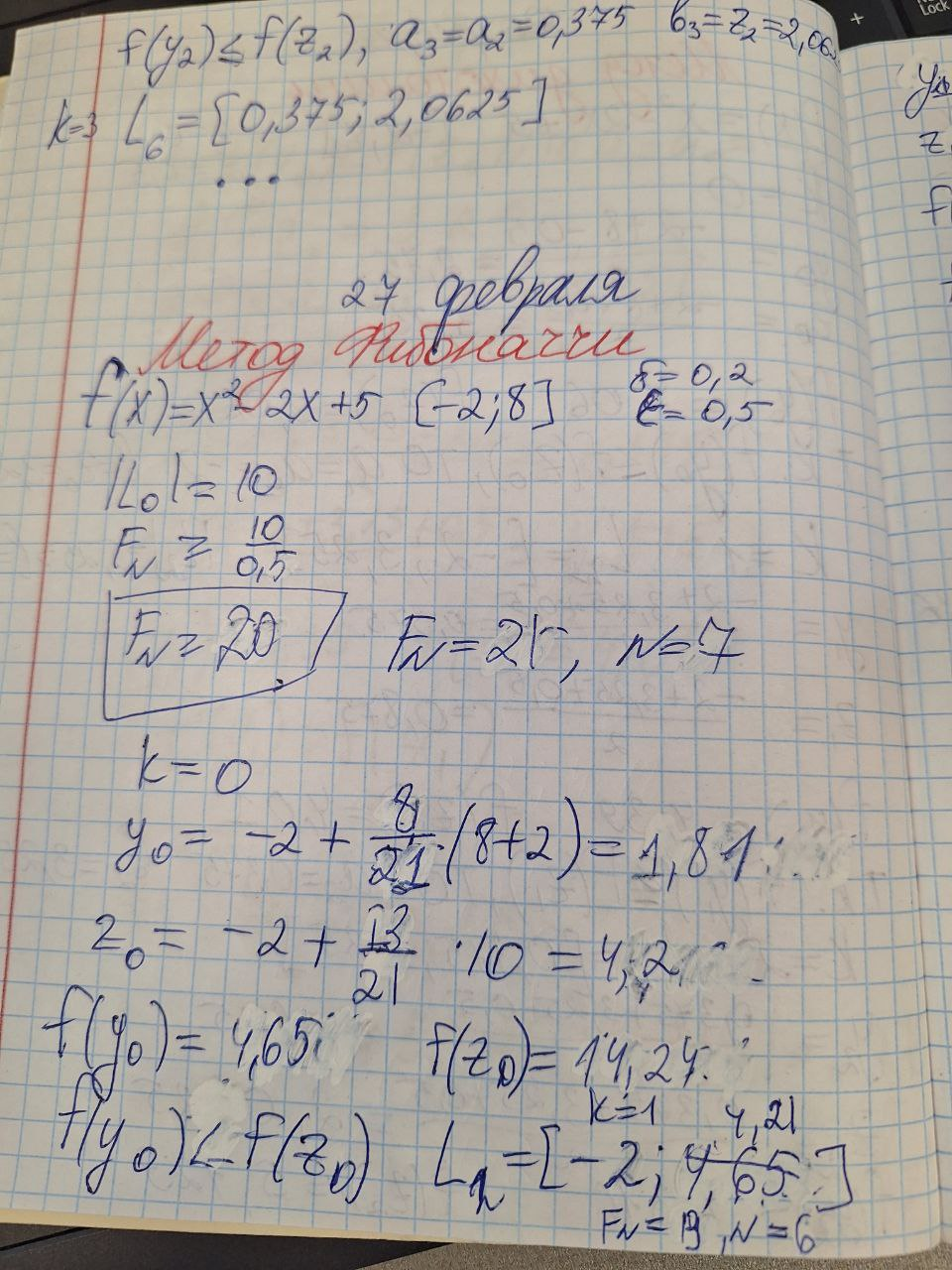
Положим yN−2=zN−2=aN−2+bN−2;

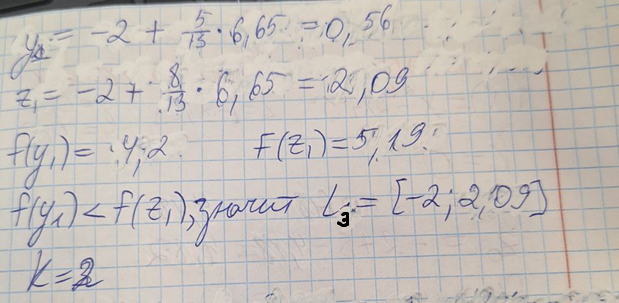
Вычислить новые точки:

* + - yN−1=yN-2
    - zN−1=yN−1+ε

Вычислить f(yN−1) и f(zN−1):

* + - **Если f(yN−1)≤f(zN−1):**  
      aN−1=aN−2​, bN−1=zN−1.
    - **Если f(yN−1)>f(zN−1):**  
      aN−1=yN−1​, bN−1=bN−2





## Программный код

def f(arg):  
 return arg \*\* 2 - 2 \* arg + 5  
  
  
def fibonacci(n):  
 if n<0:  
 return  
 if n==0 or n==1:  
 return 1  
 fib = [1, 1]  
 for i in range(2, n+1):  
 fib.append(fib[-2] + fib[-1])  
 return fib[-1]  
  
  
def calculate\_N(a, b, l):  
 N = 0  
 right\_part = abs(b - a) / l  
 while fibonacci(N) < right\_part:  
 N += 1  
 return N  
  
  
def fibonacci\_method(a, b):  
 eps = 0.2  
 l = 0.5  
  
 N = calculate\_N(a, b, l)  
  
 k = 0  
  
 y = a + (fibonacci(N - 2) / fibonacci(N)) \* (b - a)  
 z = a + (fibonacci(N - 1) / fibonacci(N)) \* (b - a)  
  
 fy = f(y)  
 fz = f(z)  
  
 while k < N - 3:  
 if fy <= fz:  
 b = z  
 z = y  
 y = a + (fibonacci(N - k - 3) / fibonacci(N - k - 1)) \* (b - a)  
 fz = fy  
 fy = f(y)  
 else:  
 a = y  
 y = z  
 z = a + (fibonacci(N - k - 2) / fibonacci(N - 1 - k)) \* (b - a)  
 fy = fz  
 fz = f(z)  
 k += 1  
  
 y = (a + b) / 2  
 z = y + eps  
  
 if f(y) <= f(z):  
 b = z  
 else:  
 a = y  
  
 print('Числа Фибоначчи')  
 for i in range(N+1):  
 print(fibonacci(i), end=' ')  
 print()  
 print(f"{a}, {b}")  
 x\_min = (a + b) / 2  
 print(f"Минимум находится в точке {x\_min} и равен {f(x\_min)}")  
 print(f"k = {k}, N = {N}")  
 print(f"R(N) = {1 / fibonacci(N)}")  
  
  
a = -2  
b = 8  
fibonacci\_method(a, b)

## Вывод на консоль:

# Метод золотого сечения

* + 1. Задать начальный интервал неопределенности:

L0[a0, b0]

* + 1. K = 0
    2. Вычисление y0 и z0

y0 = a0 + 0.618\*(b0-a0)

z0=a0+b0-y0

4k. Вычисление f(y0), f(z0)

5k. Если f(y0) ≤ f(z0):

ak+1=ak; bk+1=zk ; yk+1=ak+1+bk+1-yk ; zk+1=yk

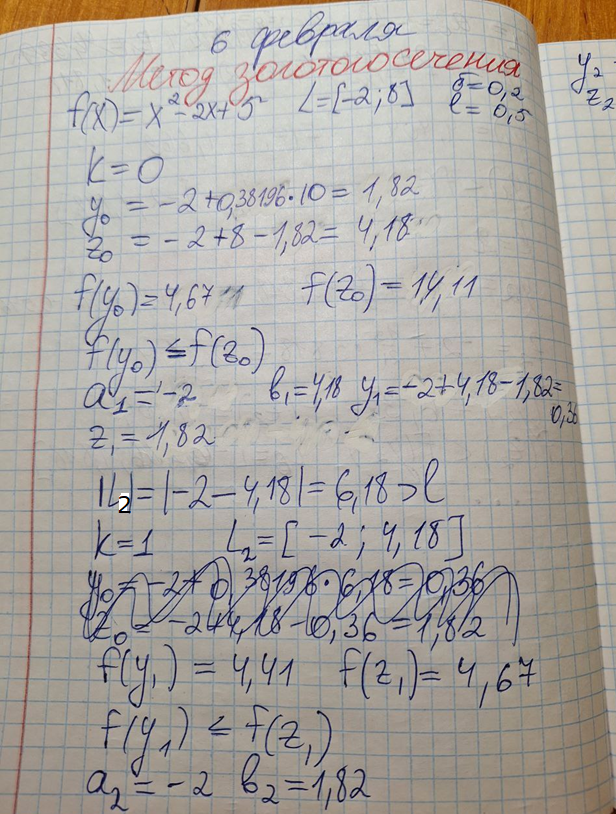
Если f(y0) > f(z0):

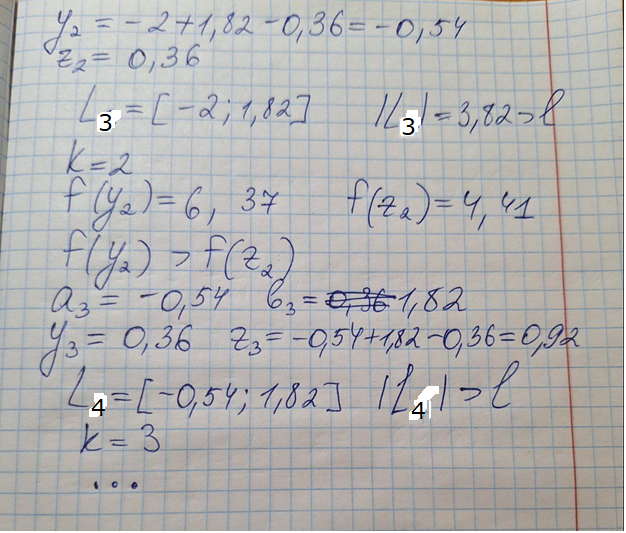
ak+1=yk; bk+1=bk ; yk+1= zk; zk+1= ak+1+bk+1-yk

6k. Если |Lk+2[ak+1;bk+1]|<l:

x\* = (ak+1+bk+1)/ 2

Иначе: k=k+1, возвращение к шагу 4k+.

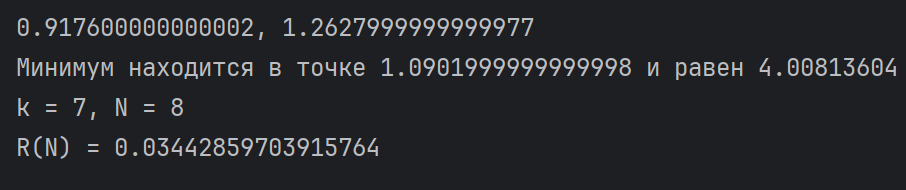




## Программный код:

PHI = 0.38196  
  
  
def f(arg):  
 return arg \*\* 2 - 2 \* arg + 5  
  
  
def golden\_section(a, b):  
 l = 0.5  
 k = 0  
 y = a + PHI \* (b - a)  
 z = a + b - y  
  
 while abs(b - a) > l:  
 if f(y) <= f(z):  
 b = z  
 z = y  
 y = a + b - y  
 else:  
 a = y  
 y = z  
 z = a + b - z  
  
 k += 1  
  
 N = k + 1  
  
 print(f"{a}, {b}")  
 x\_min = (a + b) / 2  
 print(f"Минимум находится в точке {x\_min} и равен {f(x\_min)}")  
 print(f"k = {k}, N = {N}")  
 print(f"R(N) = {0.618 \*\* (N - 1)}")  
  
  
golden\_section(-2, 8)

## Вывод на консоль:



# Анализ методов

Метод дихотомии основан на последовательном делении интервала поиска пополам. На каждой итерации вычисляются две точки, симметрично расположенные относительно середины интервала, и в зависимости от значений функции в этих точках интервал сужается. Этот метод отличается простотой реализации и устойчивостью, однако требует большего количества вычислений функции по сравнению с другими методами, что делает его менее эффективным в плане вычислительных затрат.

Метод Фибоначчи использует числа Фибоначчи для определения точек на каждой итерации. Интервал сужается пропорционально этим числам, что обеспечивает высокую скорость сходимости и минимальное количество итераций для достижения заданной точности. Однако метод требует либо заранее известного количества итераций, либо вычисления чисел Фибоначчи, что может быть неудобно в некоторых случаях. Несмотря на это, метод Фибоначчи является одним из самых эффективных с точки зрения скорости сходимости.

Метод золотого сечения основан на делении интервала в пропорции золотого сечения, которая составляет приблизительно 0.618. На каждой итерации интервал сужается в фиксированной пропорции, что обеспечивает высокую скорость сходимости. Этот метод не требует предварительных вычислений, в отличие от метода Фибоначчи, что делает его более универсальным и удобным для практического использования. Однако по количеству итераций он немного уступает методу Фибоначчи.

# Выводы

На основе проведенного анализа можно сделать следующие выводы. Метод Фибоначчи является наиболее эффективным с точки зрения количества итераций и скорости сходимости, что делает его предпочтительным выбором для задач, где критично количество вычислений функции. Метод золотого сечения представляет собой хороший компромисс между простотой реализации и эффективностью, что делает его универсальным инструментом для большинства практических задач. Метод дихотомии, несмотря на свою простоту, менее эффективен по сравнению с другими методами и требует большего количества вычислений функции.