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 № 1**

**по дисциплине  
  «Методы оптимизации»**

Выполнил студент группы 35/2                                              \_\_\_\_           А.В. Акулов

Направление подготовки  02.03.03  Математическое обеспечение и администрирование информационных систем

Курс    3

Краснодар

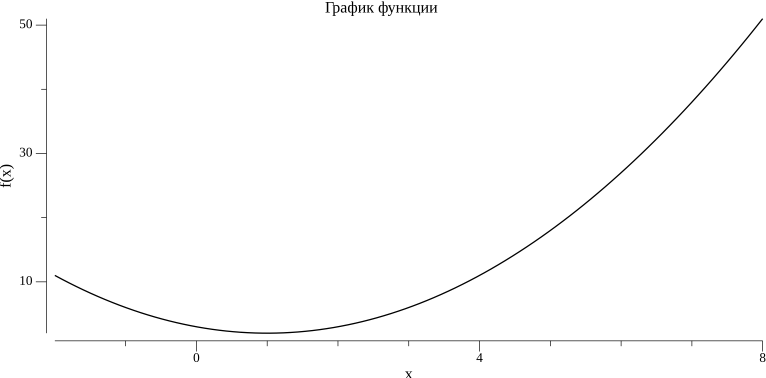
Постановка задачи

Найти минимум функции:

f(x) = x2-2\*x+3

на заданном отрезке [-2;8] методами: дихотомии, Фибоначчи, золотого сечения.

График функции представлен на рисунке ниже.

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.

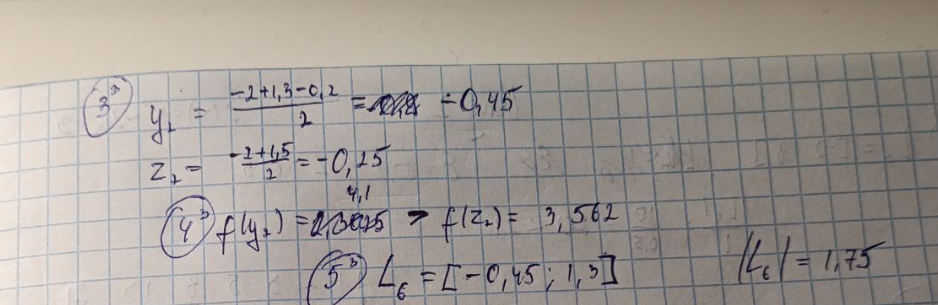
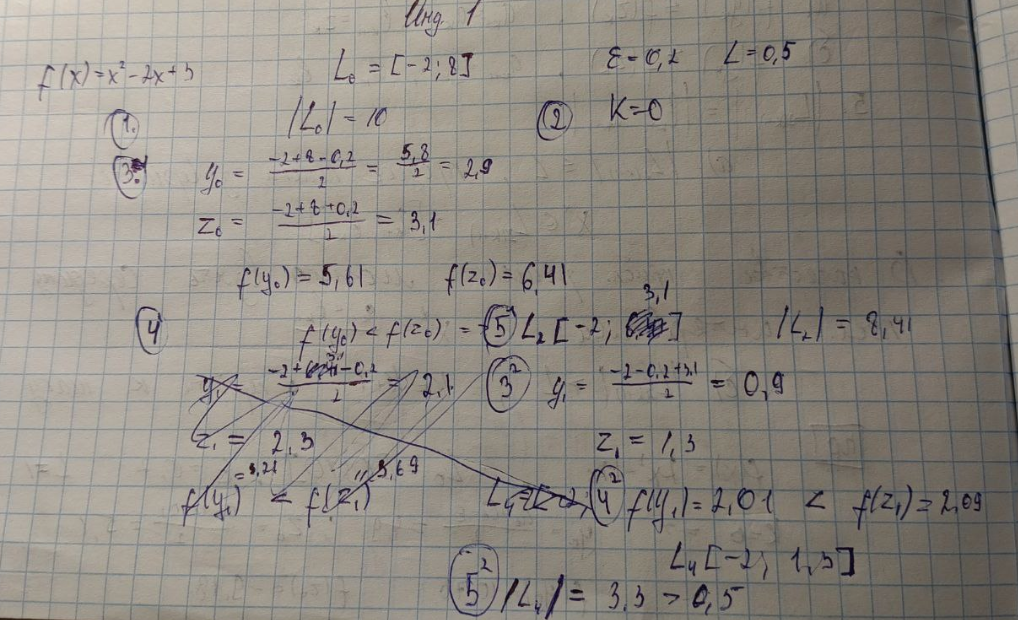
**Шаг 5.** Вычислить длину нового интервала:

∣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.

Рукописный вариант:

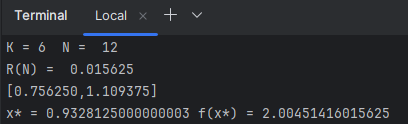


Код программы:

func dihotomoia(a, b float64) float64 {  
 eps := 0.2  
 l := 0.5  
 length := b - a  
 var y, z float64  
 k := 0  
 for float64(length) > l {  
 y = (a + b - eps) / 2  
 z = (a + b + eps) / 2  
 if f(y) <= f(z) {  
 b = z  
 } else {  
 a = y  
 }  
 length = b - a  
 k++  
 }  
 fmt.Println("K =", k, " N = ", 2\*k)  
 fmt.Println("R(N) =", 1/math.Pow(2, float64(k)))  
 fmt.Printf("[%f,%f]\n", a, b)  
 res := (a + b) / 2  
 fmt.Println("x\* =", res, "f(x\*) =", f(res))  
 return (a + b) / 2  
}

func main() {  
 dihotomoia(-2, 8)

}

Вывод:  


**Алгоритм 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:

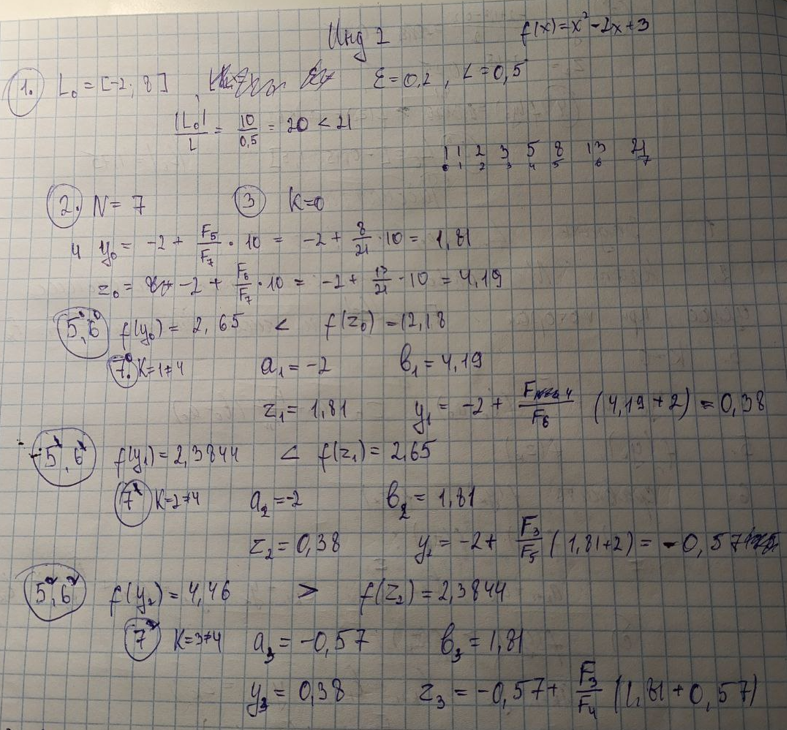
* + - yN−1=yN-2
    - zN−1=yN−1+ε

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

В качестве приближённого решения можно взять середину последнего интервала: x∗=(aN-1+bN-1)/ 2

Рукописный вариант:



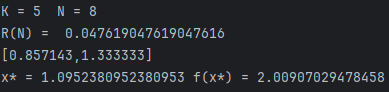
Код программы:

func fibonacci(n int) int {  
 if n <= 1 {  
 return n  
 }  
 return fibonacci(n-1) + fibonacci(n-2)  
}  
  
func fibonacciSearch(a0, b0 float64) float64 {  
 eps := 0.2  
 l := 0.5  
 L0 := b0 - a0  
  
 N := 0  
 for {  
 fib := fibonacci(N)  
 if float64(fib) >= L0/l {  
 break  
 }  
 N++  
 }  
 k := 0  
 a := a0  
 b := b0  
 var y, z float64  
  
 y = a + (float64(fibonacci(N-2))/float64(fibonacci(N)))\*(b-a)  
 z = a + (float64(fibonacci(N-1))/float64(fibonacci(N)))\*(b-a)  
  
 fy := f(y)  
 fz := f(z)  
  
 for k < N-3 {  
 if fy <= fz {  
 b = z  
 z = y  
 y = a + (float64(fibonacci(N-k-3))/float64(fibonacci(N-k-1)))\*(b-a)  
 fz = fy  
 fy = f(y)  
 } else {  
 a = y  
 y = z  
 z = a + (float64(fibonacci(N-k-2))/float64(fibonacci(N-1-k)))\*(b-a)  
 fy = fz  
 fz = f(z)  
 }  
 k++  
 }  
  
 y = (a + b) / 2  
 z = y + eps  
  
 fy = f(y)  
 fz = f(z)  
  
 if fy <= fz {  
 b = z  
 } else {  
 a = y  
 }  
 fmt.Println("K =", k, " N =", N)  
 fmt.Println("R(N) =", 1/float64(fibonacci(N)))  
 fmt.Printf("[%f,%f]\n", a, b)  
 res := (a + b) / 2  
 fmt.Println("x\* =", res, "f(x\*) =", f(res))  
 return (a + b) / 2  
}

func main() {  
 fibonacciSearch(-2, 8)

}

Вывод:



**Алгоритм золотого сечения**

* + 1. L0[a0,b0], l
    2. K = 0
    3. 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

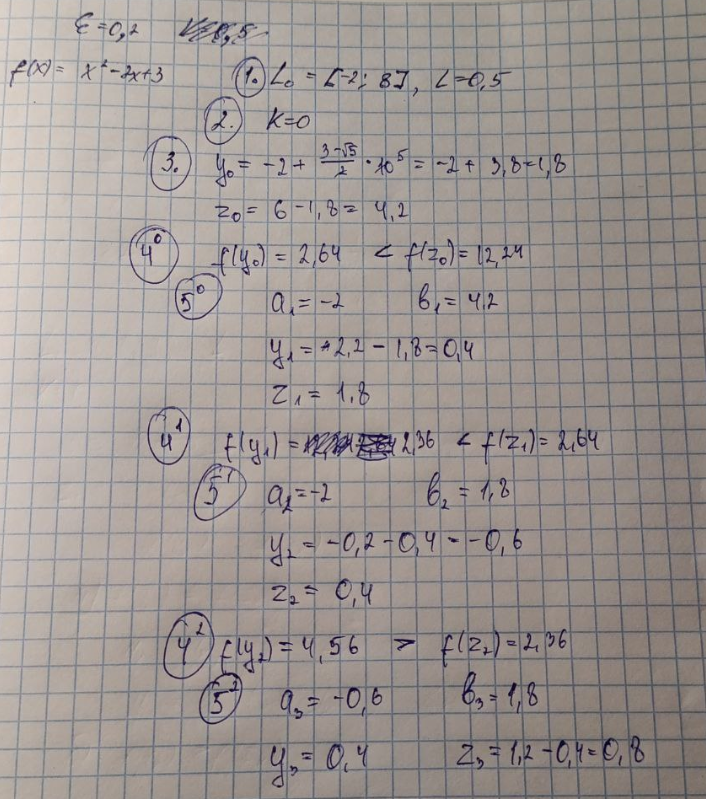
Если f(y0) > f(z0):

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+.

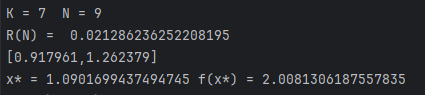
Рукописный вариант:

Код:

func goldenSection(a0, b0 float64) {  
 k := 0  
 y0 := a0 + (3-math.Sqrt(5))/2\*(b0-a0)  
 z0 := a0 + b0 - y0  
 length := b0 - a0  
 for length > 0.5 {  
 fy := f(y0)  
 fz := f(z0)  
 if fy <= fz {  
 b0 = z0  
 z0 = y0  
 y0 = a0 + b0 - y0  
 } else {  
 a0 = y0  
 y0 = z0  
 z0 = a0 + b0 - z0  
 }  
 k++  
 length = b0 - a0  
 }  
 N := k + 2  
 RN := math.Pow(0.618, float64(N-1))  
 fmt.Println("K =", k, " N =", N)  
 fmt.Println("R(N) =", RN)  
 fmt.Printf("[%f,%f]\n", a0, b0)  
 res := (a0 + b0) / 2  
 fmt.Println("x\* =", res, "f(x\*) =", f(res))  
}  
  
func main() {  
 goldenSection(-2, 8)

}

Вывод:



**Анализ**

1. Точность:

Метод дихотомии демонстрирует наивысшую точность (самый узкий интервал и минимальное значение функции), благодаря коэффициенту сокращения R(N)=1/2N/2​.

Метод Фибоначчи имеет наибольший интервал и наименее точное значение f(x∗), что связано с меньшим количеством итераций и вычислений.

Метод золотого сечения занимает промежуточное положение по точности.

1. Вычислительная стоимость:

Метод Фибоначчи требует наименьшего числа вычислений (N=8), так как использует адаптивное сокращение интервала.

Метод дихотомии самый затратный (N=12) из-за двух вычислений функции на каждой итерации.

Метод золотого сечения (N=9) ближе к Фибоначчи, но требует чуть больше вычислений из-за фиксированного коэффициента

**Выводы:**

Метод дихотомии — если критична точность, а вычислительные ресурсы не ограничены.

Метод Фибоначчи — для задач с ограниченным бюджетом вычислений, где допустима умеренная погрешность.

Метод золотого сечения — оптимален для баланса между точностью и скоростью, особенно при работе с унимодальными функциями.