**ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

(СПбГУТ)

ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ СЕТЕЙ И СИСТЕМ **(ИКСС)**

КАФЕДРА ПРОГРАММНОЙ ИНЖЕНЕРИИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ **(ПИ И ВТ)**

Отчет

По практической (лабораторной) работе №3

Дисциплина: «Алгоритмические основы программной инженерии»

Вариант №17

Выполнил: Студент группы ИКПИ-95

Новиков С.C.

Подпись \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Принял: Преподаватель:

Ерофеев С.А.

Подпись \_\_\_\_\_\_\_\_\_\_\_\_\_\_

2020г

Оглавление

Цель работы 3

Таблица идентификаторов 3

Диапазон переменных 3

Ресурсоемкость 3

[Таблица отлова ошибок](#_Toc34827270) 4

[Блок схема](#_Toc34827271) 4

[Проверка на работоспособность](#_Toc34827272) 5  
Анализ алгоритмов..................................................................................................6  
Метод секущих в профилировщике.......................................................................7  
UML- диаграмма sequence......................................................................................8

[Код на языке Golang](#_Toc34827273) 9

Вывод ....................12

# 

# Цель работы

Оценить сложность алгоритмов программы, которая решает кубические уравнения вида ax^3+bx^2+cx+d 3 методами (секущих, касательных и комбинированным) в 3 потока, коэффициенты которого вводит пользователь вручную.

# Таблица идентификаторов

|  |  |  |
| --- | --- | --- |
| N | Идентификатор | Назначение |
| 1 | var a, b, c, d float64 | Коэффициенты кубического уравнения |
| 2 | var a1, b1 float64 | Начальные приближения |
| 3 | var E = 0.01 | Погрешность |
| 4 | var xN float64 | Результат |
| 5 | var n = 0 | Счетчик итераций |
| 6 | var y float64 | Вспомогательная переменная |
| 7 | chan1 := make(chan float64, 3) | Буферизированный канал типа float64 для 3 значений |
| 9 | var err error | Информация об ошибках |
| 10 | var a, b float64 | Вспомогательные переменные |
| 11 | var x float64 | Вспомогательная переменная |

# Диапазон переменных

**float64**: представляет число с плавающей точкой от

-1.797693134862315708145274237317043567981e+308 до 1.797693134862315708145274237317043567981e+308 и занимает 8 байт.

**int64**: представляет целое число от –9 223 372 036 854 775 808 до 9 223 372 036 854 775 807 и занимает 8 байт.

# Ресурсоемкость

**error** занимает в маяти 16 байт. Всего переменных 1. В итоге 16 байт.

**float64** занимает в памяти 8 байт. Всего переменных 11. В итоге 88 байт.

**int64** занимает в памяти 8 байт. Всего переменных 1. В итоге 8 байт.

**chan** занимает в маяти 8 байт. Всего переменных 1. В итоге 8 байт.

Всего: 120 байт.

# Таблица отлова ошибок

|  |  |  |
| --- | --- | --- |
| Ошибка | Информационное сообщение | Реализация |
| Переполнение и ввод недопустимых аргументов, например, букв. | Неправильный формат данных! | \_, err := fmt.Scan(...)  if err != nil {  fmt.Println("Неправильный формат данных!")  os.Exit(0)  } |
| Пределы переменных | Не получилось посчитать, возможно мы вышли за допустимые пределы | if math.IsNaN(xN) {  fmt.Println("Не получилось посчитать, возможно мы вышли за допустимые пределы")  } |

# Блок схема

Начало

Инициализация переменных

Считывание значений a,b,c,d и проверка на корректность (тип данных)

Ошибка

Ошибка

Считывание значений x1, x2, E и проверка на корректность (тип данных)

Ошибка

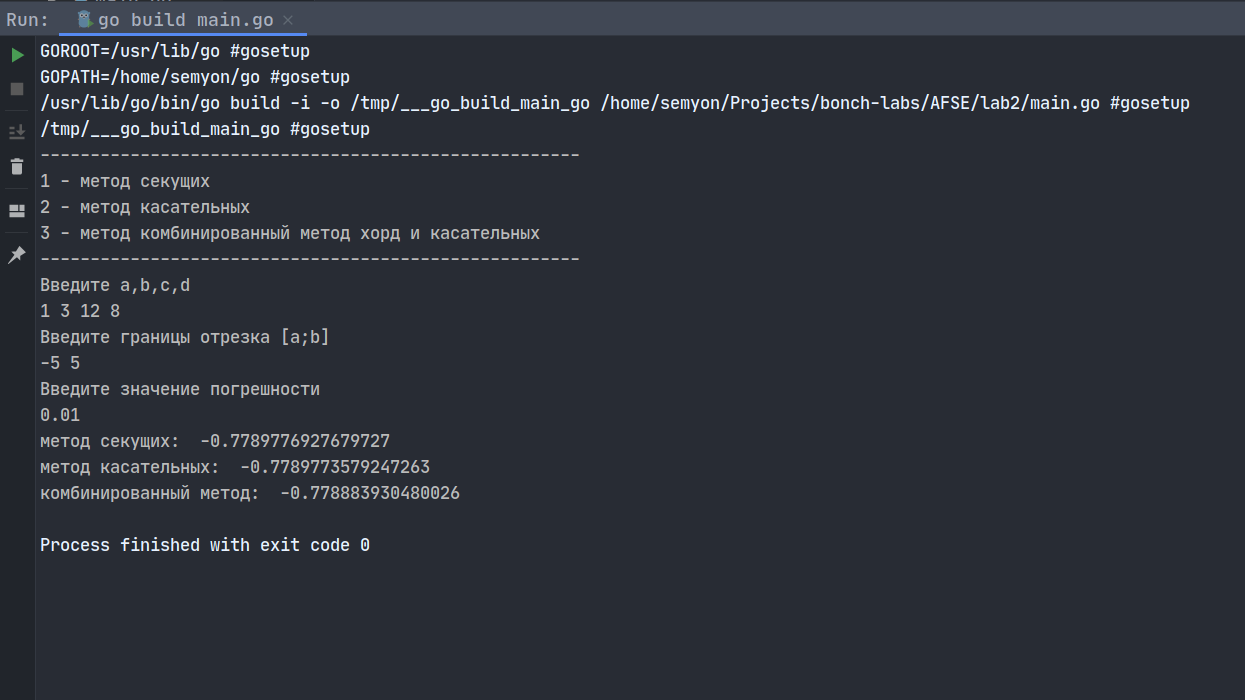
Решение уравнения 3 способами в 3 потока

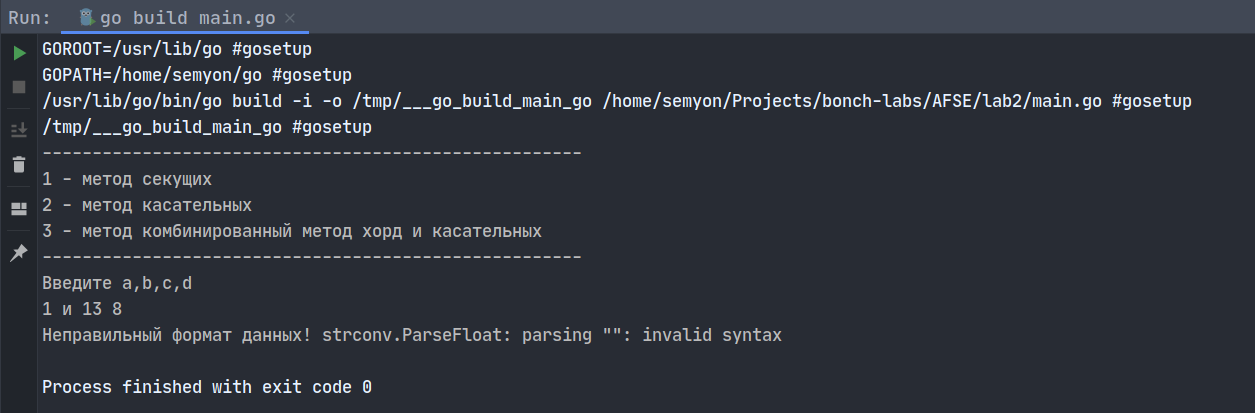
Проверка результата

Вывод результата

Конец

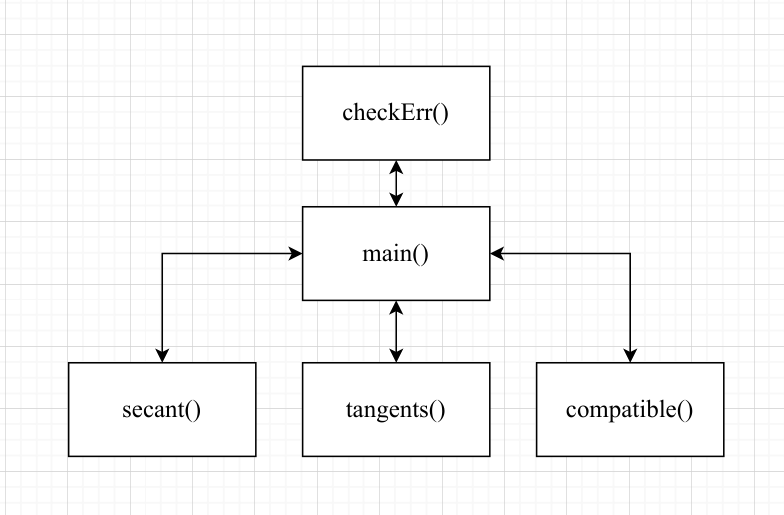
# Проверка на работоспособность





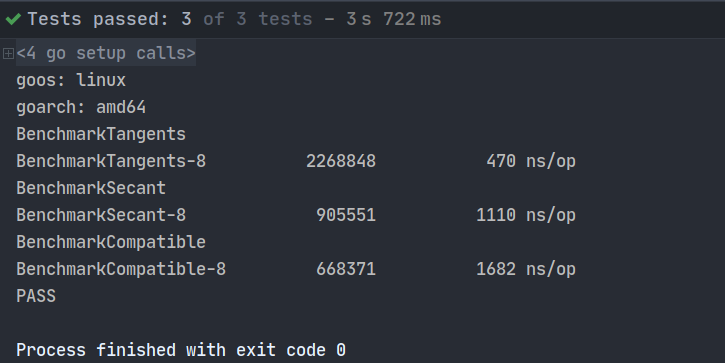
# 

# Блок схема вызовов функций



# Анализ алгоритмов

По скорости:



Потребление памяти: 0.671234 MB

Все алгоритмы имеют сложность t(n) так как содержат итерационный процесс в котором выполняется постоянное количество «C» операций, не зависящее от n. Каждый алгоритм содержит только один цикл.

# Метод секущих в профилировщике

# UML- диаграмма sequence

Решение системы уравнений

Главный поток

# 

Ввод данных

Проверка входных данных

Сообщение об ошибке

Чтение границ

Проверка входных данных

Сообщение об ошибке

Передача и

Вызов функций для решения

Сообщение об ошибке при попытке решения

Вывод результата

# Код на языке Golang

Файл main.go

package main

import (

"errors"

"fmt"

"math"

"os"

)

var a, b, c, d float64

var a1, b1 float64

var E = 0.01 // погрешность

// уравнение

func f(x float64) float64 {

return a\*math.Pow(x, 3) + b\*math.Pow(x, 2) + c\*x + d

}

// производная от уравнения

func fp1(x float64) float64 {

return 3\*a\*math.Pow(x, 2) + 2\*b\*x + c

}

// производная от уравнения

func fp2(x float64) float64 {

return 6\*a\*x + 2\*b

}

func main() {

fmt.Println("------------------------------------------------------")

fmt.Println("1 - метод секущих")

fmt.Println("2 - метод касательных")

fmt.Println("3 - метод комбинированный метод хорд и касательных")

fmt.Println("------------------------------------------------------")

fmt.Println("Введите a,b,c,d")

\_, err := fmt.Scan(&a, &b, &c, &d)

checkErr(err, "Неправильный формат данных!")

if a == 0 && b == 0 && c == 0 {

if d == 0 {

fmt.Println("x - любое")

os.Exit(0)

} else if d != 0 {

fmt.Println("нет решений")

os.Exit(0)

}

}

fmt.Println("Введите границы отрезка [a;b]")

\_, err = fmt.Scan(&a1, &b1)

checkErr(err, "Неправильный формат отрезка!")

if a1 > b1 {

checkErr(errors.New("неверно указан интервал поиска"), "")

}

fmt.Println("Введите значение погрешности")

\_, err = fmt.Scan(&E)

checkErr(err, "Неправильный формат погрешности!")

if E < 0 {

checkErr(errors.New("значение погрешности должно быть положительным"), "")

}

chan1 := make(chan float64, 3)

go secant(chan1) // метод секущих

fmt.Println("метод секущих: ", <-chan1)

go tangents(chan1) // метод касательныx

fmt.Println("метод касательныx: ", <-chan1)

go compatible(chan1) // комбинированный метод

fmt.Println("комбинированный метод: ", <-chan1)

}

// метод касательных (Ньютона)

func tangents(c chan float64) {

var x float64

x0 := x

f0 := f(x0)

x = x0

for math.Abs(f(x)) > E {

x -= f0 / fp1(x)

f0 = f(x)

}

c <- x

}

// метод секущих

func secant(c chan float64) {

var y, xN float64

var n = 0

for {

n++

y = xN

xN = b1 - ((b1-a1)/(f(b1)-f(a1)))\*f(b1)

a1 = b1

b1 = xN

if !(math.Abs(y-xN) >= E) {

break

}

}

if math.IsNaN(xN) {

fmt.Println("Не получилось посчитать, возможно мы вышли за допустимые пределы")

}

c <- xN

}

// метод комбинированный метод хорд и касательных

func compatible(c chan float64) { // функция вычисляет по методу хорд и касательных

var a, b float64

a = a1

b = b1

var k int

for ; math.Abs(b-a) > E\*2; { //Построение хорд и касательных продолжается до достижения необходимой точности решения е

if f(a)\*fp2(a) < 0 {

a += (b - a) / (f(a) - f(b)) \* f(a)

} else {

a -= f(a) / fp1(a)

}

if f(b)\*fp2(b) < 0 {

b += (a - b) / (f(b) - f(a)) \* f(b)

} else {

b -= f(b) / fp1(b)

}

k++

}

c <- (a + b) / 2.0

}

func checkErr(err error, toPrint string) {

if err != nil {

fmt.Println(toPrint, err)

os.Exit(0)

}

}

# Вывод

Поставлена задача измерить алгоритмы по скорости и памяти решения кубического уравнения в 3 потока 3 разными методами. Данная задача реализована на языке Go. В этом алгоритме реализовано решение, постановленной задачи с контролем входных данных. В контроле входных данных реализована проверка данных на некорректный ввод и переполнение переменной.