МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА» (БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**Лабораторная работа № 6**по дисциплине: «Вычислительная математика»

Выполнил: ст. группы ПВ-211

Медведев Д.С.

Проверила:

Бондаренко Т.В.

Белгород 2023 г.

Одномерная минимизация функции

# Вариант 8

**Цель работы:** изучить методы нахождения приближенного решения задачи одномерной минимизации функции одной переменной, и получить практические навыки их применения.

**Ход работы**

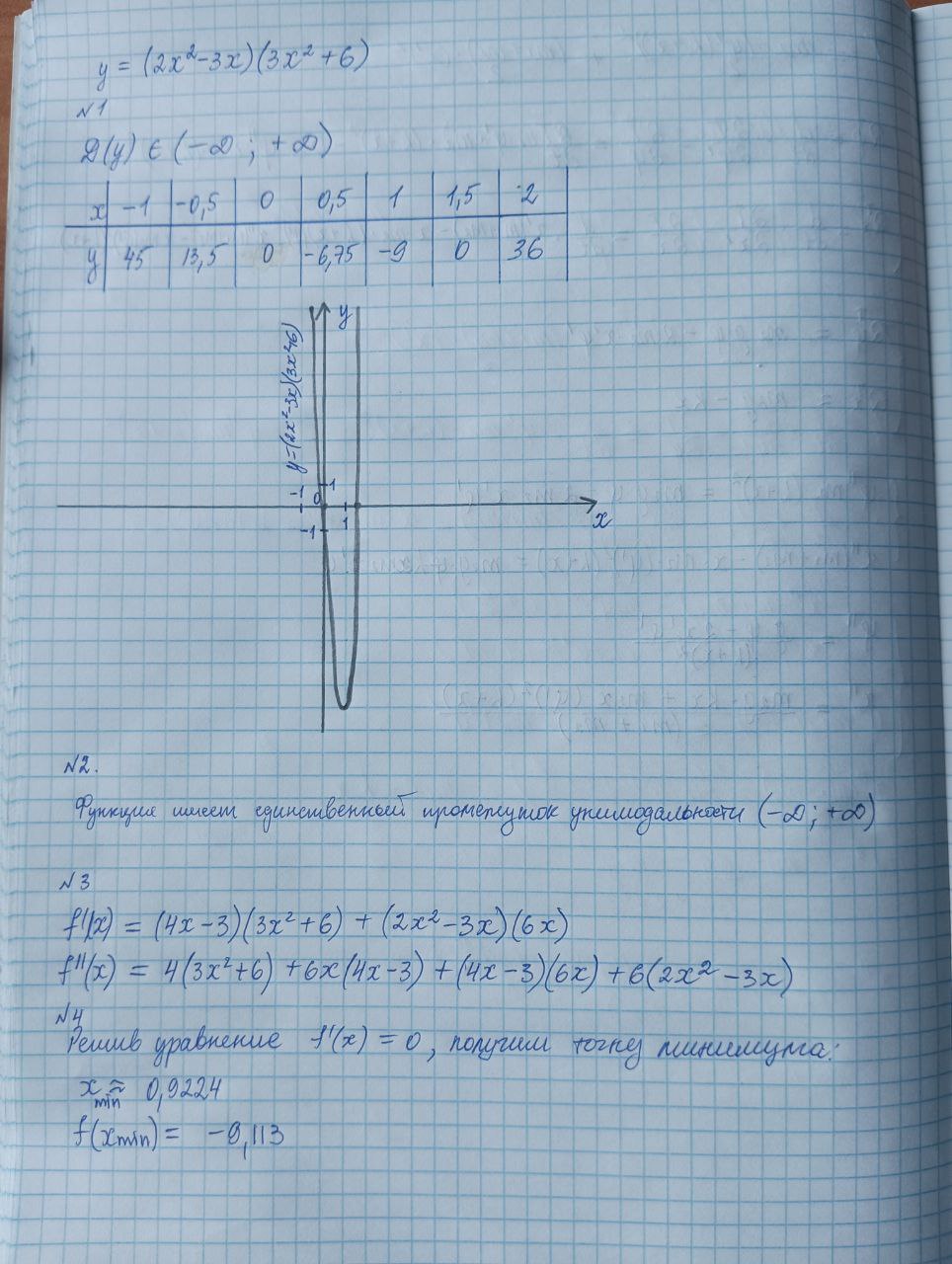
****

1. Найти область определения заданной функции у = f(x) и построить её график, используя равномерную сетку значений хi (шаг сетки выбрать самостоятельно).

2. Найти промежутки унимодальности функцииу=f(x), используя построенный график.

3. Найти первую y ́=f ́(x) и вторую y ́ ́= f ́ ́ (x) производные заданной функции у = f(x).

4. Найти точное решение задачи одномерной минимизации ― минимум функции у = f(x), точку хТ, и минимальное значение функции



5. Найти приближенное решение задачи одномерной

минимизации, точку такую, что вручную, используя численные методы одномерной минимизации:

* метод оптимального поиска;





xmin = 0.92

f(xmin) = -9.113

* метод деления отрезка пополам;  
  

xmin = 0.9187

f(xmin) = -9.1129

* метод, основанный на использовании чисел Фибоначчи  
  

xmin = 0.90725

f(xmin) = -9.1126

с точностью ε =0,01.

Необходимые параметры методов выбрать самостоятельно.

Подробно «вручную» достаточно выполнить только первый шаг численного метода решения.

Окончательный результат вычислений может быть получен с помощью приложения MS Excel.

6. Определить абсолютную Δ и относительную δ погрешность решения задачи одномерной минимизации для каждого из используемых численных методов. Представить полученные результаты в виде таблицы (табл. 6.1).

|  |  |  |  |
| --- | --- | --- | --- |
| Погрешность | Метод оптимального поиска | Метод деления отрезка пополам | Метод чисел Фибоначчи |
| Δ | -0,0024 | -0,0037047 | -0,01515 |
| δ | 0,00260191 | 0,00401636 | 0,01642454 |

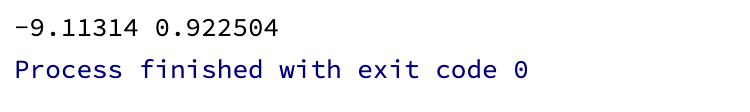
7. Описать в модуле функции, которые возвращают приближенные значения минимума функции у = f(x) для заданного промежутка унимодальности с заданной точностью ε каждым из рассмотренных численных методов: метод оптимального поиска; метод, основанный на использовании чисел Фибоначчи; метод деления отрезка пополам.

#include **<iostream>**#include **"cmath"**#include **"vector"  
  
using namespace** std;  
  
**typedef float** func(**float** x);  
  
**struct** Segment {  
 **float** l;  
 **float** r;  
};  
  
**float** taskFunction(**float** x) {  
 **return** (2 \* pow(x, 2) - 3 \* x) \* (3 \* pow(x, 2) + 6);  
}  
  
**float** findMinFunctionValue(func f, Segment segment, **float** eps, **float** &xMinValue) {  
 **float** minValueArgument = segment.l;  
 **float** minFunctionValue = f(segment.l);  
  
 **for** (**float** x = segment.l; x <= segment.r; x += eps) {  
 **if** (f(x) < minFunctionValue) {  
 minFunctionValue = f(x);  
 minValueArgument = x;  
 }  
 }  
  
 xMinValue = minValueArgument;  
  
 **return** minFunctionValue;  
}  
  
**float** findMinFunctionValueDivisionByTwo(func f, Segment segment, **float** eps, **float** &xMinValue) {  
  
 **float** precision = abs(segment.r - segment.l);  
  
 **while** (precision > eps) {  
 **float** alpha = (segment.l + segment.r) / 2 - (segment.r - segment.l) / 4;  
 **float** betta = (segment.l + segment.r) / 2 + (segment.r - segment.l) / 4;  
  
 **if** (f(alpha) >= f(betta)) {  
 segment.l = alpha;  
 } **else** {  
 segment.r = betta;  
 }  
  
 precision = abs(segment.r - segment.l);  
 }  
  
 xMinValue = segment.l;  
 **return** f(segment.l);  
}  
  
**float** findMinFunctionValueFibonacci(func f, Segment segment, **float** eps, **float** &xMinValue) {  
 **int** N = 100;  
  
 vector<**float**> fibonacciValues**{**1, 1**}**;  
 **for** (**int** i = 2; i <= N; i++) {  
 fibonacciValues.push\_back(fibonacciValues[fibonacciValues.size() - 2] +  
 fibonacciValues[fibonacciValues.size() - 1]);  
 }  
  
 **float** precision = abs(segment.r - segment.l);  
  
 **int** step = 1;  
 **while** (precision > eps && step < N - 1) {  
 **float** delta = segment.r - segment.l;  
 **int** fibonacciMainIndex = N - step;  
 **float** alpha = segment.l + fibonacciValues[fibonacciMainIndex - 1] /  
 fibonacciValues[fibonacciMainIndex + 1] \* delta;  
 **float** betta = segment.l + fibonacciValues[fibonacciMainIndex] /  
 fibonacciValues[fibonacciMainIndex + 1] \* delta;  
  
 **if** (f(alpha) >= f(betta)) {  
 segment.l = alpha;  
 } **else** {  
 segment.r = betta;  
 }  
  
 precision = abs(segment.r - segment.l);  
 step++;  
 }  
  
 xMinValue = segment.l;  
 **return** f(segment.l);  
}

8. Составить программу для вычисления приближенного решения задачи одномерной минимизации для заданного варианта задания с использованием функций, описанных в модуле.

**int** main() {  
 **float** xMinValue;  
 cout << findMinFunctionValueFibonacci(taskFunction, {0, 2}, 0.00001, xMinValue) << **" "** << xMinValue;  
}

**Вывод программы:**



**Вывод:** в ходе лабораторной работы мы изучили методы нахождения приближенного решения задачи одномерной минимизации функции одной переменной, и получили практические навыки их применения.