**Федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»**

Лабораторная работа №2

по вычислительной математике

Вариант: 21

Выполнил: Носов Михаил Александрович

Группа: Р3212

Принял(а)

Преподаватель: Малышева Татьяна Алексеевна

САНКТ-ПЕТЕРБУРГ

2021

Цель работы

Цель работы: создать программу, реализующую поиск корней нелинейных уравнений.

1 Все численные методы должны быть реализованы в виде отдельных подпрограмм или классов.

2 Пользователь выбирает уравнение, корень/корни которого требуется вычислить (3-5 функций, в том числе и трансцендентные), из тех, которые предлагает про-грамма.

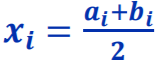
3 Предусмотреть ввод исходных данных (границы интервала/начальное приближение к корню и погрешность вычисления) из файла или с клавиатуры по выбору конечного пользователя.

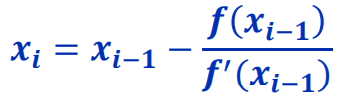
4 Выполнить верификацию исходных данных. Для метода половинного деления (метода хорд) анализировать наличие корня на введенном интервале. Для метода Ньютона (метода секущих) – выбор начального приближения (а или b). Для метода простой итерации – достаточное условие сходимости метода. Программа должна реагировать на некорректные введенные данные.

5 Предусмотреть вывод результатов (найденный корень уравнения, значение функции в корне, число итераций) в файл или на экран по выбору конечного пользователя.

6 Организовать вывод графика функции, график должен полностью отображать весь исследуемый интервал (с запасом).

Рабочие формулы

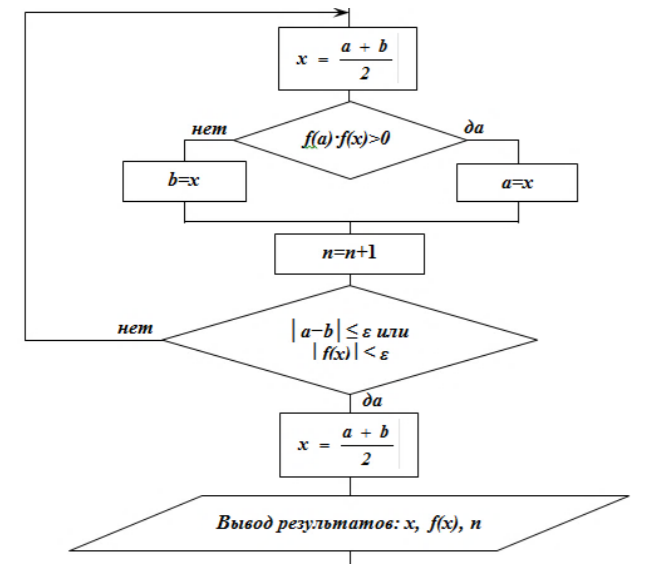
Метод половинного деления 

Метод Ньютона 

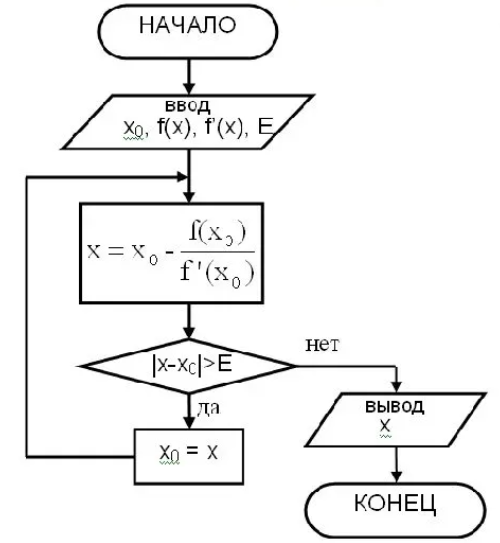
Метод простых итераций 

Блок-схемы методов

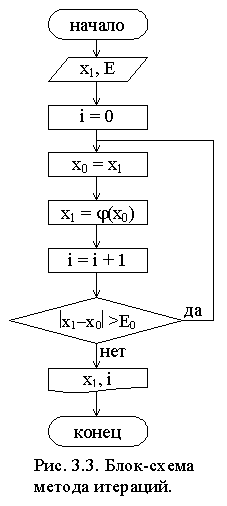
Метод половинного деления



Метод Ньютона



Метод простых итераций



Таблицы результатов

Уточнение корня уравнения методом половинного деления

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № шага | a | b | x | f(a) | f(b) | f(x) | |a-b| |
| 1 | 3.5 | 4.5 | 4.0 | -10.724 | 6.630 | -4.87 | 1.0 |
| 2 | 4.0 | 4.5 | 4.25 | -4.87 | 6.630 | 0.098 | 0.5 |
| 3 | 4.0 | 4.25 | 4.125 | -4.87 | 0.098 | -2.571 | 0.25 |
| 4 | 4.125 | 4.25 | 4.187 | -2.571 | 0.098 | -1.284 | 0.125 |
| 5 | 4.187 | 4.25 | 4.218 | -1.284 | 0.098 | -0.604 | 0.062 |
| 6 | 4.218 | 4.25 | 4.234 | -0.604 | 0.098 | -0.256 | 0.031 |
| 7 | 4.234 | 4.25 | 4.242 | -0.256 | 0.098 | -0.079 | 0.015 |
| 8 | 4.242 | 4.25 | 4.246 | -0.079 | 0.098 | 0.009 | 0.007 |

Уточнение корня уравнения методом Ньютона

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № итерации | *xk* | *f*(*xk* ) | *f '*(*xk*) | *xk*+1 | │*xk* − *xk*+1│ |
| 1 | -1,113 | -12,927 | 28,554 | -0,660 | 0,453 |
| 2 | -0,660 | -2,629 | 17,272 | -0,508 | 0,152 |
| 3 | -0,508 | -0,257 | 13,927 | -0,490 | 0,018 |
| 4 | -0,490 | -0,004 | 13,536 | -0,489 | 0,000 |

Уточнение корня уравнения методом простой итерации

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № итерации | *xk* | *f*(*xk* ) | *xk*+1 |  | │*xk* − *xk*+1│ |
| 1 | 1,305 | -0,260 | 1,268 | -8,605 | 0,030 |
| 2 | 1,268 | 0,055 | 1,276 | -8,465 | 0,006 |
| 3 | 1,276 | -0,011 | 1,274 | -8,495 | 0,001 |
| 4 | 1,274 | 0,002 | 1,275 | -8,489 | 0,000 |

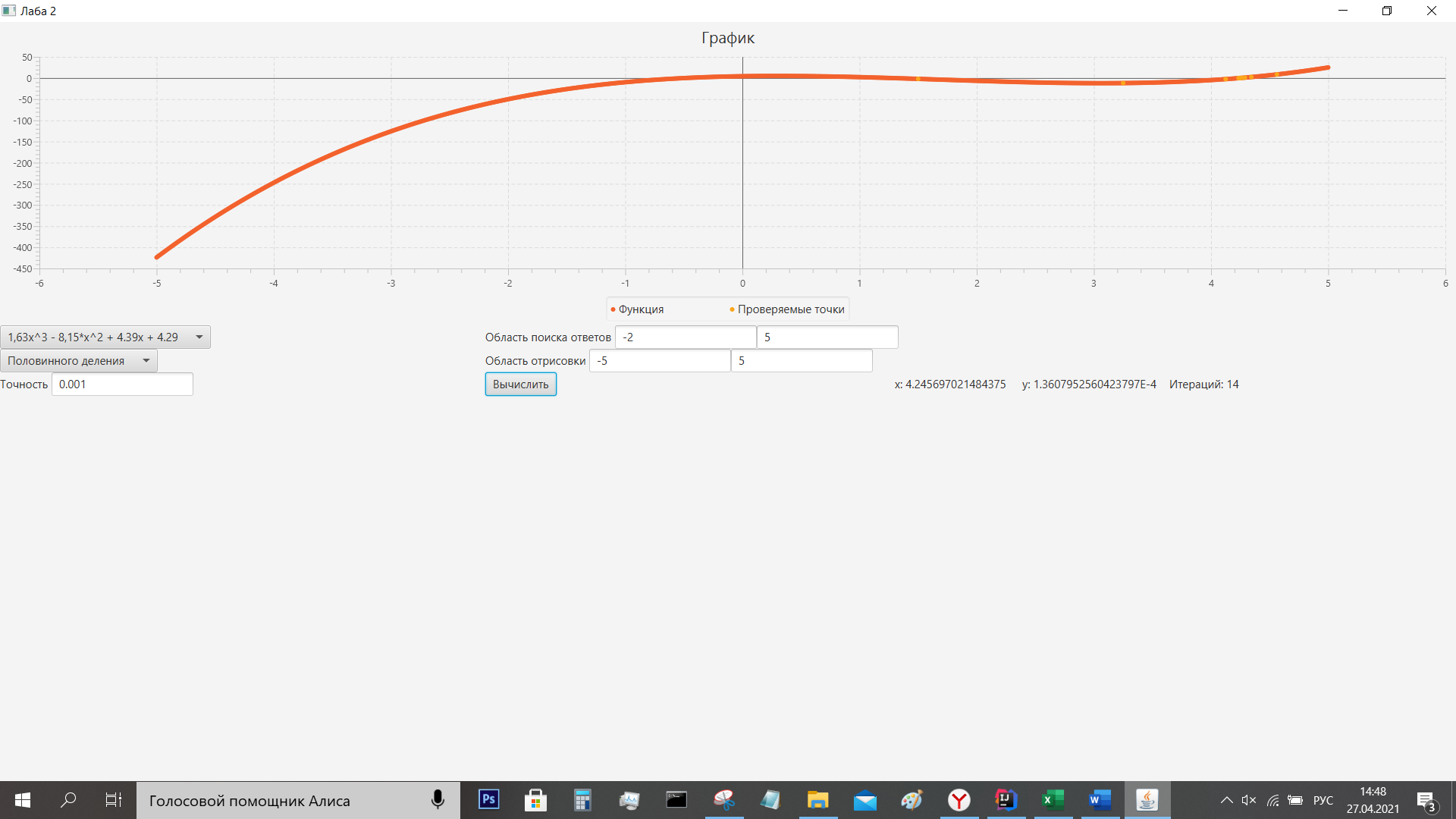
Листинг программы

public double halfs(double a, double b, ObservableList<XYChart.Data> datas2, double l, double m, double n) throws Exception { // Метод половинного деления

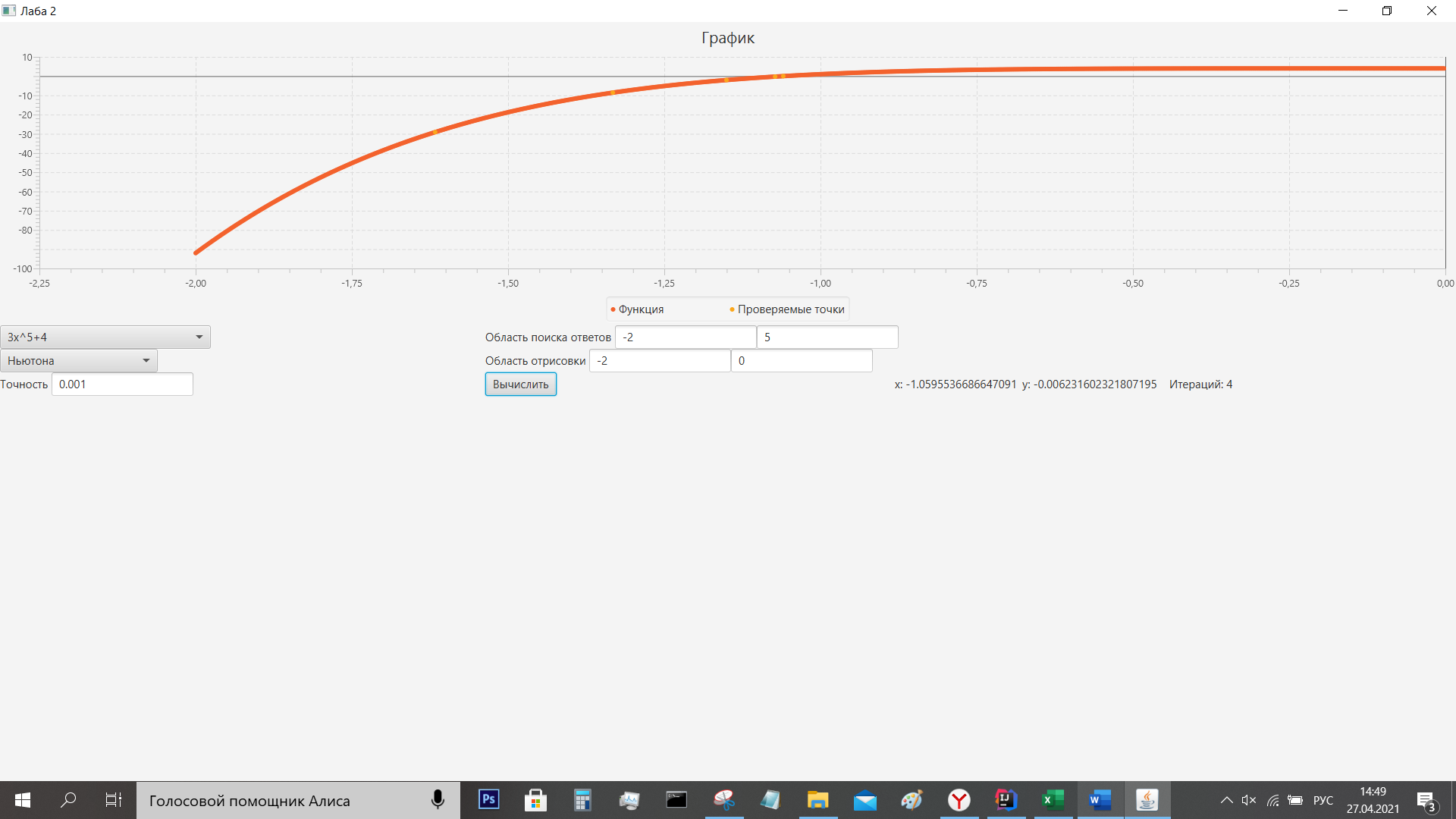
double x = (a+b)/2;  
 if((x > n)&&(x < m)) {  
 datas2.add(new XYChart.Data(x, function(x)));  
 }  
 i = 0;  
 for(;*mod*(function(x)) > l;){  
 i++;  
 if(function(a)\*function(x) > 0){  
 if(x == a)throw new Exception();  
 a = x;  
 }else{  
 if(x == b)throw new Exception();  
 b = x;  
 }  
 x = (a+b)/2;  
 if((x > n)&&(x < m)){  
 datas2.add(new XYChart.Data(x, function(x)));  
 }  
 }  
 return x;  
}  
  
public double nuton(double a, double b, ObservableList<XYChart.Data> datas2, double l, double m, double n) throws Exception { //Метод Ньютона  
 double x;  
 if(function(a)/functionA(a) > 0){  
 x = b - function(b)/functionA(b);  
 }else {  
 x = a - function(a) / functionA(a);  
 }  
 if((x > n)&&(x < m)) {  
 datas2.add(new XYChart.Data(x, function(x)));  
 }  
 i = 0;  
 for(;*mod*(function(x)/(functionA(x))) > l;){  
 i++;  
 x = x - function(x)/functionA(x);  
 if((x > n)&&(x < m)){  
 datas2.add(new XYChart.Data(x, function(x)));  
 }  
 }  
 if(x < a || x > b){  
 throw new Exception();  
 }  
 return x;  
}  
  
public double simple(double a, double b, ObservableList<XYChart.Data> datas2, double l, double m, double n) throws Exception { //Метод простых итераций  
 double r = -1/max(functionA(a), functionA(b));  
 double x;  
 if(max(functionA(a), functionA(b)) == functionA(a)) {  
 x = a + r \* function(a);  
 }else{  
 x = b + r \* function(b);  
 }  
 if((x > n)&&(x < m)) {  
 datas2.add(new XYChart.Data(x, function(x)));  
 }  
 i = 0;  
 for(;((*mod*(*mod*(x - *mod*(x + r\*function(x)))) > l/20) && (*mod*(r\*function(x)) > l/20) && (x != (x + r\*function(x) + r\*function(x + r\*function(x)))));){  
 i++;  
 x = x + r\*function(x);  
 if((x > n)&&(x < m)){  
 datas2.add(new XYChart.Data(x, function(x)));  
 }  
 }  
 if(*mod*(function(x)) > l\*40){  
 throw new Exception();}  
 return x;  
}

Примеры работы программы

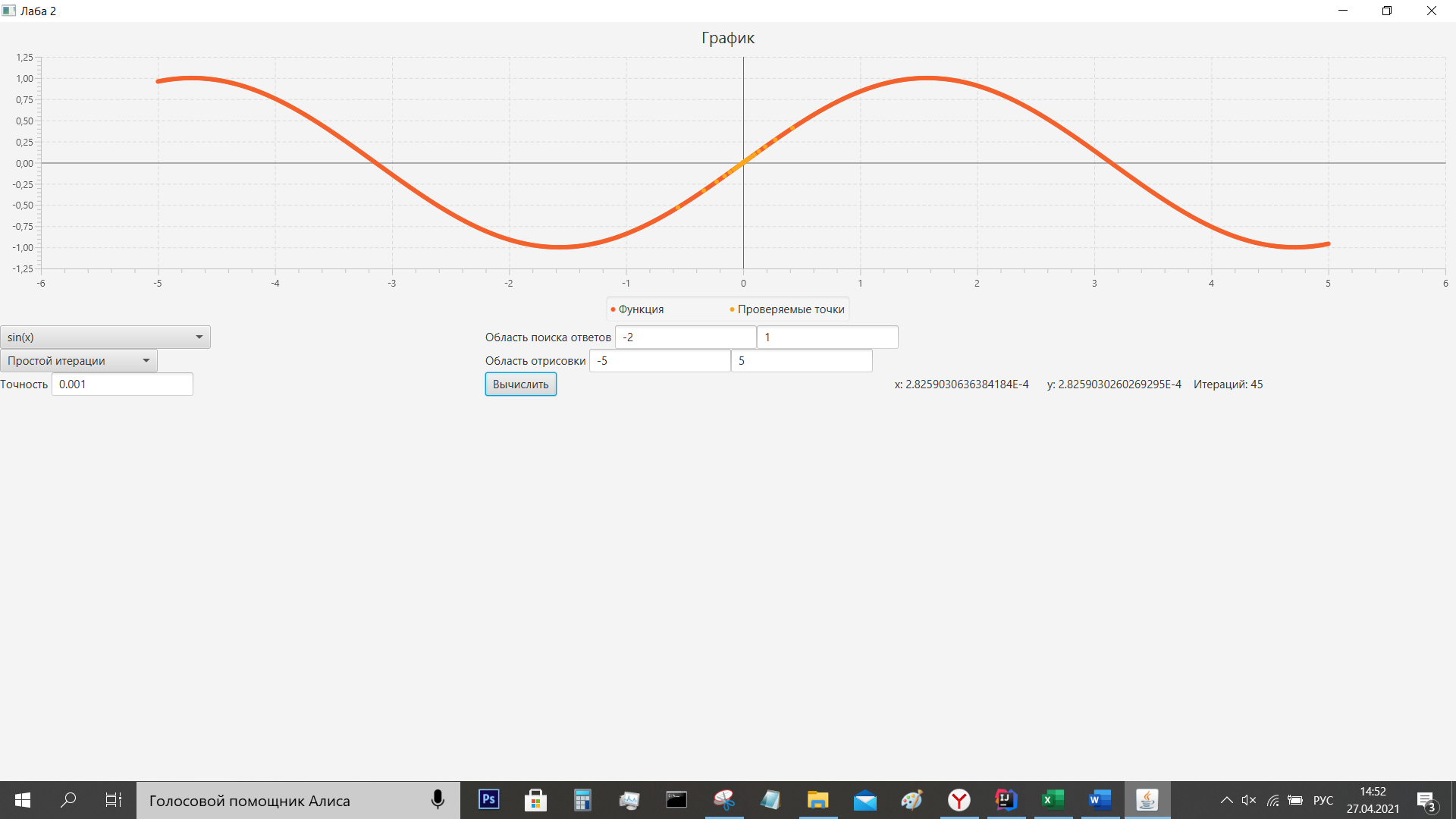
Пример №1



Пример №2



Пример №3



Вывод: методы половинного деления и Ньютона работают хорошо и достаточно надёжно, в то время как методу простых итераций не хватает условия, что значения не повторяются с некоторой периодичностью, что иногда случается с тригонометрическими функциями.