Министерство образования Республики Беларусь

Учреждение образования “Брестский Государственный Технический Университет”

Кафедра ИИТ

Лабораторная работа №1

По дисциплине Вычислительная математика

«Решение нелинейных уравнений»

Выполнил: Студент 1 курса

Группы АС-59

Быбко Т.А.

Проверил: Пролиско Е.Е.

Брест 2021

Вариант 46

Цель: применить на практике один из итерационных методов и написать программу его выполнения.

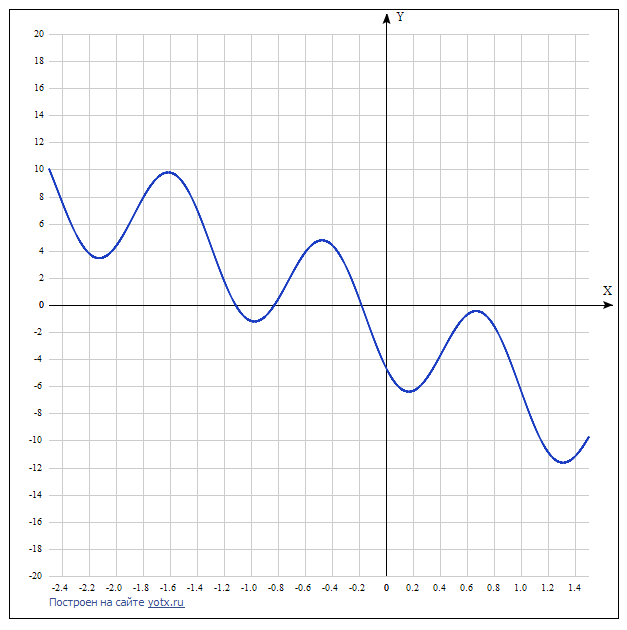
Условие задания: найти наименьший корень функции

y = -4.6\*x-1.1\*exp(-1.7\*x-4.1)+4.2\*sin(5.5\*x+4)-1.5

на интервале [-2.5, 1.5] с точностью 10^(-6). Решение получить методом Ньютона.

Ход работы:

Построим график функции:



Отсюда видно, что наименьший корень функции принадлежит интервалу [-1.2;-1], этот интервал и будет интервалом локализации корня.

Найдём производные от данной по условию функции, для того, чтобы применить метод Ньютона:

Первая производная: y’ = -4.6 + 1.87 \* exp(-1.7 \* x - 4.1) + 23.1 \* cos(5.5 \* x + 4);

Вторая производная: y’’ = -3.179 \* exp(-1.7 \* x - 4.1) - 127.05 \* sin(5.5 \* x + 4);

Далее напишем программу на С++, которая найдёт наименьший корень уравнения:

#include <iostream>

#include <iomanip> //для работы с setprecision()

#include <cmath>

using namespace std;

double Fy(double);

double dFy(double);

double d2Fy(double);

int main()

{

setlocale(LC\_ALL, "Russian");

double E = 0.000001; //точность

cout << setprecision(10); /\*для того, чтобы результаты выводились с нужной точностью

(здесь используется 10, а не 7, чтобы конечный ответ вывелся более точно (до значения 10, конечный ответ выводился только с 5 знаками после запятой)) \*/

double a = -1.2, b = -1.0; /\*Интервал локализации корня (тот интервал, на котором лежит минимальный корень функции)\*/

cout << "Интервал локализации корня: [" << a << ", " << b << "]" << endl;

cout << "Значения функции в крайних точках интервала a и b соответственно: " << Fy(a) << " " << Fy(b) << endl;

cout << "Значения первой производной в крайних точках интервала a и b соответственно: " << dFy(a) << " " << dFy(b) << endl;

cout << "Значения второй производной в крайних точках интервала a и b соответственно: " << d2Fy(a) << " " << d2Fy(b) << endl;

double m1 = fabs(dFy(b)); /\*абсолютный минимум функции на интервале [a,b] (понадобится для условия остановки цикла)\*/

double x0 = 0;

for (double i = a; i < b; i += 0.001) /\*цикл для поиска начального приблежения согласно условию F(x0)\*(F(x0))'' > 0 \*/

{

if (d2Fy(i) \* Fy(i) > 0)

{

x0 = i;

break;

}

}

cout << "Начальное приближение: " << x0 << endl;

double x = 0; // Переменная, которая будет нужна для цикла ниже

int i = 0; /\*Кол-во итераций (начинается с нуля, т.к. в цикле ниже, в самой последней итерации цикл добавит к i лишнюю единицу) \*/

for (i = 0; fabs(Fy(x)) / m1 > E; i++) //fabs(Fy(x))/m1 < E это условие остановки

{

x = x0 - (Fy(x0) / dFy(x0));

x0 = x;

}

/\*Переменные в цикле выше обозначают следующее: x это Xi, x0 это X(i-1), где i - кол-во итераций. \*/

cout << "Наименьший корень функции: " << x << endl;

cout << "Количество итераций: " << i << endl;

cout << "Невязка функции: " << fabs(Fy(x)) << endl;

return 0;

}

double Fy(double x) /\*функция, которая вычисляет данную по условию функцию по передаваемому в функцию x \*/

{

double y = -4.6 \* x - 1.1 \* exp(-1.7 \* x - 4.1) + 4.2 \* sin(5.5 \* x + 4) - 1.5;

return y;

}

double dFy(double x) /\*функция, которая вычисляет первую производную данную по условию функцию по передаваемому в функцию x \*/

{

double dy = -4.6 + 1.87 \* exp(-1.7 \* x - 4.1) + 23.1 \* cos(5.5 \* x + 4);

return dy;

}

double d2Fy(double x) /\*функция, которая вычисляет вторую производную данную по условию функцию по передаваемому в функцию x \*/

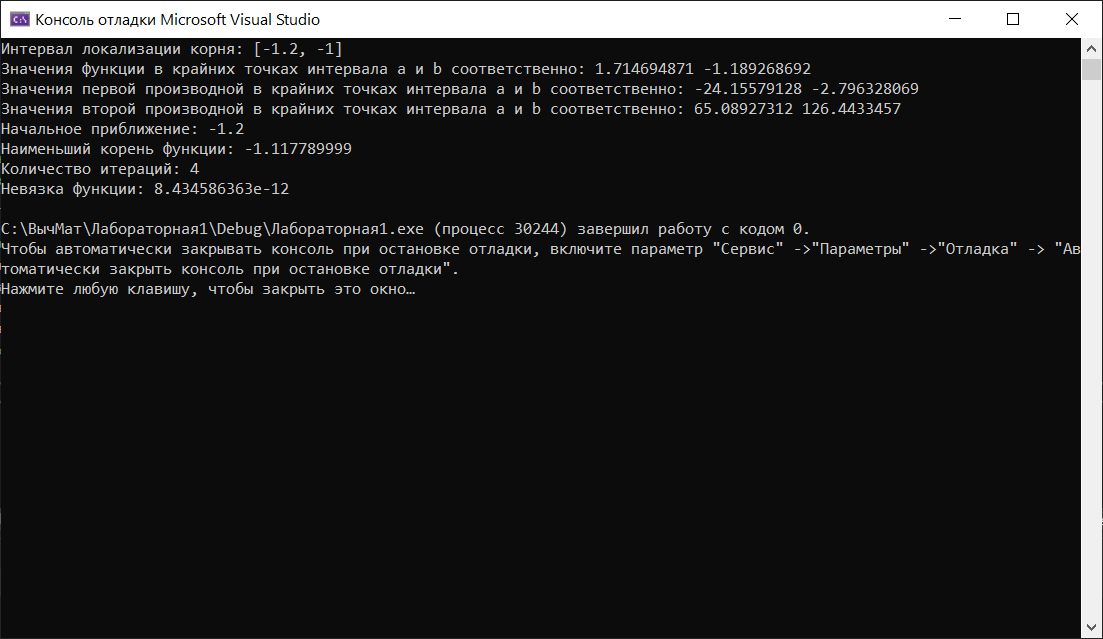
{

double d2y = -3.179 \* exp(-1.7 \* x - 4.1) - 127.05 \* sin(5.5 \* x + 4);

return d2y;

}

Результат выполнения программы:



Вывод: задание было выполнено с помощью метода Ньютона. Для этого пришлось находить производные сложной функции, благо как это делать ещё осталось в памяти с прошлого семестра. Сложнейшим этапом стало изучение непосредственно метода Ньютона, ибо некоторое время у меня были проблемы с тем, чтобы его понять. Тем не менее, благодаря примерам, разобраться в нём получилось. Составление же самой программы не было сложным, здесь разве что пришлось узнать о функции setprecision() и адекватно реализовать все математические операции, необходимые для выполнения метода Ньютона.