|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | |
| МИНОБРНАУКИ РОССИИ | | | |
| 1. Федеральное государственное бюджетное образовательное учреждение 2. высшего профессионального образования 3. **"Московский технологический университет"** 4. МИРЭА | | | |
| Институт Кибернетики | | |  |
| Кафедра программного обеспечения систем радиоэлектронной аппаратуры при АО «Концерн радиостроения «ВЕГА» | | |  |

|  |  |
| --- | --- |
| **Отчет по лабораторной работе №1** | |
| **по дисциплине** | |
| **«** Численные методы **»** | |
| **Вариант 5** | |
| Студент 3-го курса  группы КМБО-02-16 | Карабанов В.В. |
| Преподаватель | Даева С.Г. |
| Рецензент |  |
|  |  |

|  |  |  |
| --- | --- | --- |
| Работа представлена к защите | «\_\_»\_\_\_\_\_\_\_2018 г. |  |
|  |  |  |
| «Допущен к защите» | «\_\_»\_\_\_\_\_\_\_2018 г. |  |

Москва 2018

Содержание

Задание №1 3

Теоретическая часть 3

Практическая часть 4

Задание №2 5

Теоретическая часть 5

Практическая часть 6

Задание №3 7

Теоретическая часть 8

Практическая часть 10

Приложения 12

# Задание №1

1. Определить, какое равенство точнее:
2. Округлить сомнительные цифры числа , оставив верные знаки, и определить абсолютную погрешность результата.
3. Найти предельные абсолютные и относительные погрешности числа , если оно имеет только верные цифры.
4. Вычислить и определить погрешность результата, если исходные числа заданы в верных знаках. Записать результат в верных знаках.

, где *a=0.562 b=0.2518 h=0.6*

## Теоретическая часть

Пусть A — точное число, a — приближенное.

Тогда число называется абсолютной погрешностью приближенного числа а.

Предельной абсолютной погрешностью числа а называется любое число не меньшее

Относительной погрешностью приближенного числа а называется число

Предельной относительной погрешностьюприближенного числа а называется любое число не меньшее

Цифра стоящая на n-ом месте в записи приближенного числа называется верной, если абсолютная погрешность этого числа не превышает десятичного разряда, выраженного n-ой цифрой данного числа.

Справедлива теорема, утверждающая, что если приближенное число *a>0* имеет *n>2* верных знаков, то относительная погрешность где — первая значащая цифра числа.

Пусть для функции известны — абсолютные погрешности аргументов. Тогда абсолютная погрешность функции

## Практическая часть

1) Для того, чтобы определить какой равенство точнее, найдем предельные относительные погрешности.

следовательно, равенство *b* точнее.

2) Запишем *а* в верных знаках, для этого найдем абсолютную погрешность: Округлим *а* до единиц, получаем: Следовательно *а* записано в верных знаках.

Ответ: *а = 11.*

3)

4) *a=0.345; b=0.2518; h=0.6;*

Тогда

Округлим *s* до тысячных: N, следовательно, округлим *s* до сотых: , следовательно, округлим *s* до десятых: , следовательно, N записано в верных знаках.

Ответ: s *= 0.6.*

# Задание №2

1. Найти границы действительных корней, используя схему Горнера. Отделить корни и уточнить каждый из них методом итерации с точностью до 0.001.
2. Отделить корни уравнения графически и уточнить один из них методом итерации с точностью до 0.001.

## Теоретическая часть

Пользуясь схемой Горнера можно вычислить границы действительных корней многочлена Если в схеме Горнера длявсе коэффициенты и то все действительные корни многочлена, если они есть, расположены не правее Для оценки нижней границы необходимо описанным выше способом найти верхнюю границу многочлена при этом нижняя граница многочлена будет равна

Для отделения корней, можно воспользоваться тем фактом, что непрерывная функция *f(x)* принимает значения разных знаков на концах отрезка [*a, b*], то внутри отрезка содержится по крайней мере один корень уравнения *f(x) = 0*. Причем тот корень будет единственным, если на отрезке существует производная *f΄(x),* и она сохраняет на нем знак.

Для уточнения корня уравнения *f(x) =* 0, где *f(x)* — непрерывная на данном отрезке функция,с заданной точностью *ε*, можно использовать метод итерации. Его суть заключается в эквивалентном преобразовании исходного уравнения к виду *x = φ(x),* выборе начального приближения и вычислением последовательности до тех пор, пока так как можно показать, чтоПри этом преобразование законно только тогда, когда *φ(x)* дифференцируема на [*a, b*] ив этом случае метод сходится независимо от выбора начального приближения.

## Практическая часть

1) Уточним корни используя схему Горнера.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 0 | 1 | -3 |
| 2 | 1 | 2 | 4 | 5 |

Корни *P(x)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 0 | 1 | 3 |
| 1 | 1 | 1 | 2 | 5 |

Отделим корни: *P(1) = -1; P(2) = 7;*

следовательно, на данном отрезке один корень.

Уточним корень методом итерации. Преобразуем исходное уравнение:

на [1, 2], следовательно, возрастает на данном отрезке, тогда:

Оценим

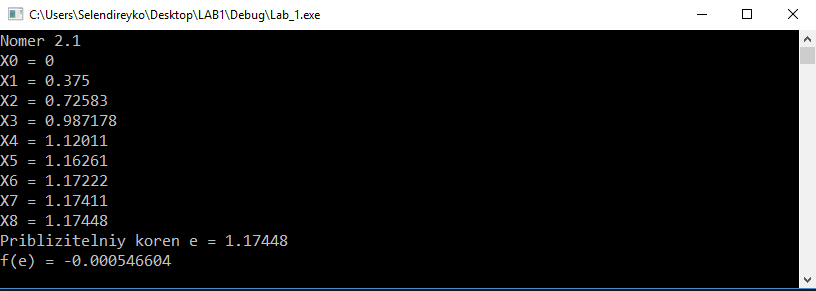
следовательно, метод итерации сходится независимо от начального приближения.

Для расчета последовательности до тех пор, пока при данных точности *ε,* функции *φ(x)* и начальном приближении была написана программа на языке “C++” в операционной системе “Windows” с использованием текстового редактора “Visual Studio 2017”.

Для данной задачи имеем: необходимую точность *ε = 0.001,* функцию В качестве начального приближения берем точку

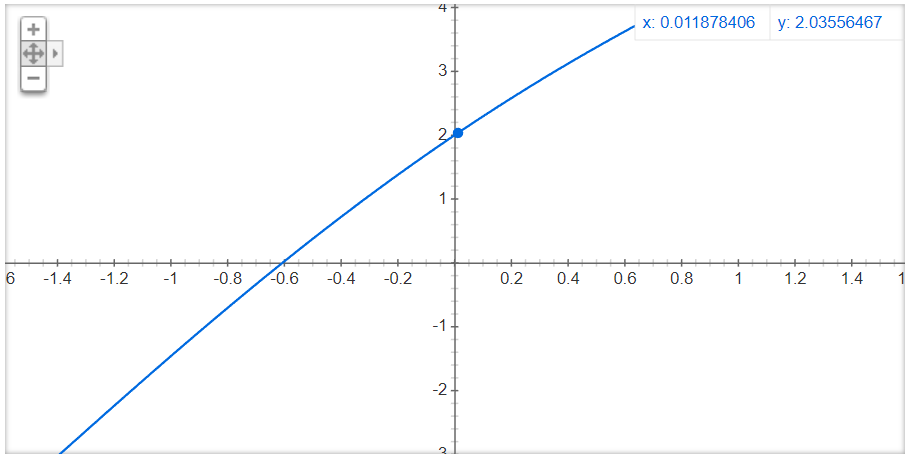
Результат: приближенный корень

Вывод программы:



2)

Графически отделим корни:



Уточним корень на отрезке [-1; 0] методом итерации. Преобразуем исходное уравнение:

на отрезке [-1; 0], следовательно, убывает на этом отрезке, тогда:

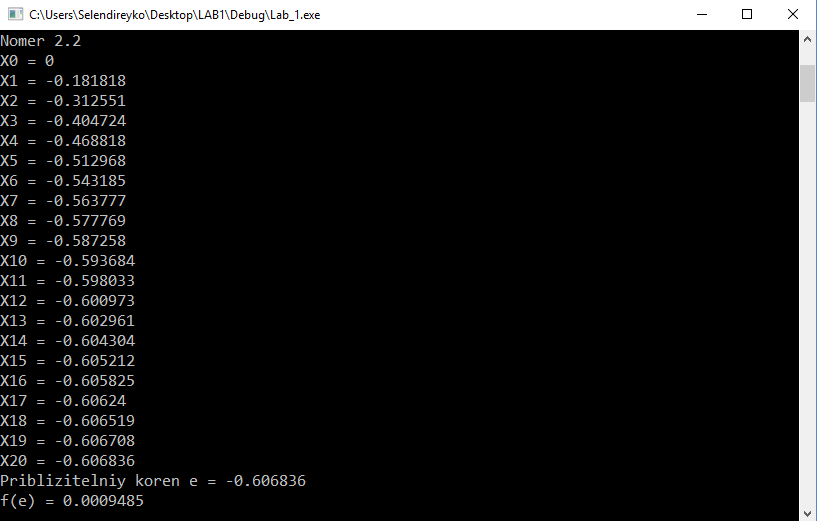
Оценим

следовательно, метод итерации сходится независимо от начального приближения.

Уточним корень с точностью *ε = 0.001,* используяфункциюВ качестве начального приближения берем точку

Результат: приближенный корень

Вывод программы:



# Задание №3

Отделить корни уравнений и уточнить по одному из них с точностью до 0.001

1. методом хорд:
2. методом касательных:
3. комбинированным методом:

## Теоретическая часть

Для уточнения корняуравнения *f(x) =* 0, где функция *f(x)* имеет непрерывные и сохраняющие знак на данном отрезке первую и вторую производные,с заданной точностью *ε*, можно использовать метод хорд. Его суть заключается в построении последовательности приближений где - точка пересечения оси абсцисс и хорды между значениями функции в точках *c* и Точка *с —* неподвижный конец отрезка [a, b], в котором знак функции совпадает со знаком второй производной. В качестве начального приближения выбирается тот конец отрезка [a, b], в котором знак функции не совпадает со знаком второй производной. Оценка погрешности метода производится по следующей формуле: где *M* и *m —* соответственномаксимальное и минимальное значениена отрезке [a, b]. Таким образом, для получения заданной точности *ε,* необходимо рассчитывать приближения до тех пор, пока не будет выполнено неравенство:

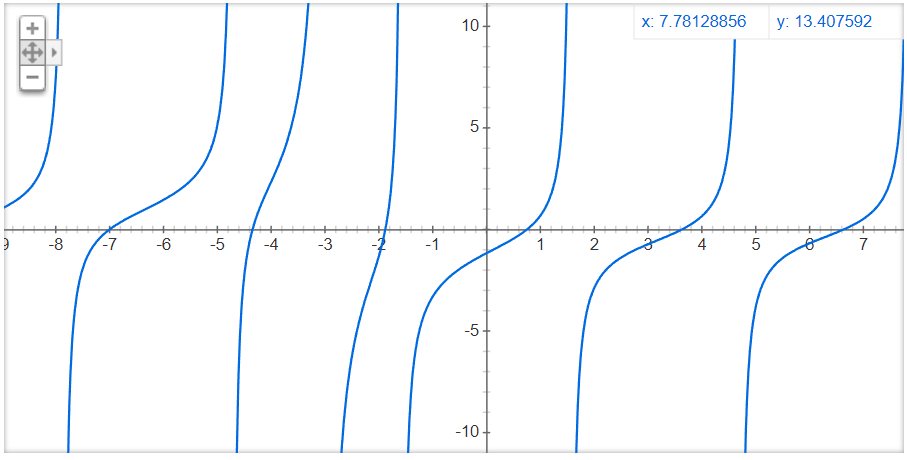
Для уточнения корня уравнения *f(x) =* 0, где функция *f(x)* имеет непрерывные и сохраняющие знак на данном отрезке первую и вторую производные, применяется метод касательных. Метод состоит в вычислении последовательности приближений Можно показать, что если функция на концах отрезка принимает разные знаки, а и отличны от нуля и сохраняют знаки, то при выборе начального приближения последовательность приближений будет сходится. Оценка погрешности метода производится по следующей формуле: Где*-* максимальное значение модуля второй производной, минимальное значение модуля первой производной на отрезке [a, b]. Таким образом, для получения заданной точности *ε,* необходимо рассчитывать приближения до тех пор, пока не будет выполнено неравенство:

Если метод хорд дает приближение по избытку или недостатку, то метод касательных — по недостатку или избытку соответственною. Таким образом для уточнения корня можно использовать оба способа одновременно. Этот метод называется комбинированным. На каждой итерации приближение, полученное методом касательных, используется в качестве неподвижного конца для метода хорд. Погрешность оценивается по следующей формуле: , где- приближения, полученные методом касательных, - приближения, полученные методом хорд.

## Практическая часть

1. 1)

Графически отделим корни:

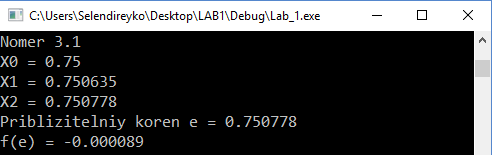


Уточним корень на отрезкеметодом хорд. Как видно из графика, на данном отрезке, следовательно 1 — неподвижный конец, 0.75 начальное приближение

Рассчитаем последовательность до тех пор, пока не выполнится неравенство

Результат: приближенный корень

Вывод программы:



2)

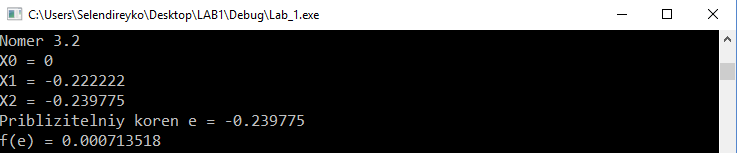
, следовательно, *f(x)* имеет только один корень.

, следовательно, корень находится на отрезке [-0.5, 0]. Уточним корень используя метод касательных.

Возьмем начальное приближение Рассчитаем последовательность до тех пор, пока не выполнится неравенство

Результат: приближенный корень

Вывод программы:



3)

следовательно, вещественная ось разбивается на три участка, где могут быть корни.

следовательно, на отрезке [-7, -6] находится один корень.

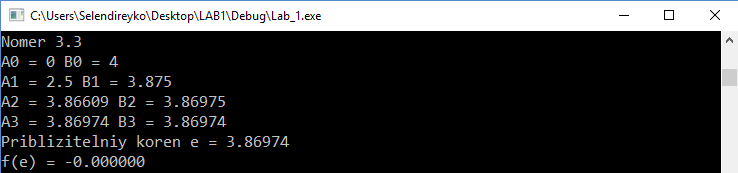
следовательно, на отрезке [-6, 0] находится один корень.

следовательно, на отрезке [0, 4] находится один корень.

Уточним корень на отрезке [0, 4] комбинированным методом. Рассчитаем последовательности до тех пор, пока не выполнится неравенство

Результат: приближенный корень

Вывод программы:



# Приложения

**Исходный код**

**Source1.cpp:**

#include <iostream>

#include <cmath>

#include "header.h"

#include <windows.h>

#define PI 3.141592

using namespace std;

double f1(double x) { return pow(x, 3) + pow(x, 2) - 3; }

double iterFunc1(double x) { return x - f1(x) / 8; }

double f2(double x) { return 3 \* x + cos(x) + 1; }

double iterFunc2(double x) { return x - f2(x) / 11; }

double f3(double x) { return tan(x) - 7 / (2 \* x + 6); }

double chordsIterFunc3(double x) { return x - f3(x)\*(x - 1) / (f3(x) - f3(1)); }

double f4(double x) { return pow(x, 3) + 3 \* pow(x, 2) + 9 \* x + 2; }

double tangentsIterFunc4(double x) { return x - f4(x) / (3 \* pow(x, 2) + 6 \* x + 9); }

double f5(double x) { return pow(x, 3) + 3 \* pow(x, 2) - 24 \* x - 10; }

double chordsIterFunc5(double x, double fixedX)

{

return x - f5(x)\*(fixedX - x) / (f5(fixedX) - f5(x));

}

double tangentsIterFunc5(double x) { return x - f5(x) / (3 \* pow(x, 2) + 6 \* x - 24); }

int main()

{

cout << "Nomer 2.1\n";

double e1 = iterMethod(0, 0.001, iterFunc1, f1);

cout << "Priblizitelniy koren e = " << e1 << endl;

cout << "f(e) = " << f1(e1) << endl;

cout << "Nomer 2.2\n";

double e2 = iterMethod(0, 0.001, iterFunc2, f2);

cout << "Priblizitelniy koren e = " << e2 << endl;

cout << "f(e) = " << f2(e2) << endl;

cout << "Nomer 3.1\n";

double e3 = iterMethod(0.75, 0.0001, chordsIterFunc3, f3);

cout << "Priblizitelniy koren e = " << e3 << endl;

printf("f(e) = %lf\n", f3(e3));

cout << "Nomer 3.2\n";

double e4 = iterMethod(0, 0.001, tangentsIterFunc4, f4);

cout << "Priblizitelniy koren e = " << e4 << endl;

cout << "f(e) = " << f4(e4) << endl;

cout << "Nomer 3.3\n";

double e5 = combMethod(0, 4, 0.001, chordsIterFunc5, tangentsIterFunc5, f5);

cout << "Priblizitelniy koren e = " << e5 << endl;

printf("f(e) = %lf\n", f5(e5));

system("pause");

return 0;

}

**Source.cpp:**

#include "header.h"

using namespace std;

double iterMethod(double firstX, double accuracy, double(\*iterFunc)(double), double(\*f)(double))

{

int i = 1;

double prevX, nextX = firstX;

cout << "X" << 0 << " = " << firstX << endl;

do

{

prevX = nextX;

nextX = iterFunc(prevX);

cout << "X" << i << " = " << nextX << endl;

i++;

} while (abs(f(nextX)) > accuracy);

return nextX;

}

double combMethod(double horda, double kasat, double accuracy,

double(\*chordsFunc)(double, double), double(\*tangentsFunc)(double), double(\*f)(double))

{

int i = 1;

cout << "A" << 0 << " = " << horda << " "

<< "B" << 0 << " = " << kasat << endl;

do

{

if (horda == kasat)return horda;

horda = chordsFunc(horda, kasat);

kasat = tangentsFunc(kasat);

cout << "A" << i << " = " << horda << " "

<< "B" << i << " = " << kasat << endl;

i++;

} while (abs(f(abs(horda + kasat) / 2.)) > accuracy);

return horda;

}

**header.h:**

#include <iostream>

#include <cmath>

double iterMethod(double firstX, double accuracy, double(\*iterFunc)(double), double(\*f)(double));

double combMethod(double A0, double B0, double accuracy,

double(\*chordsIterFunc)(double, double), double(\*tangentsFunc)(double), double(\*f)(double));