|  |  |
| --- | --- |
|  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |
| Институт Искусственного Интеллекта | | |
| Кафедра программного обеспечения систем радиоэлектронной аппаратуры | | |

|  |  |
| --- | --- |
| **Отчет по лабораторной работе №1** | |
| **по дисциплине** | |
| **«** Численные методы **»** | |
| **Вариант 34** | |
| Студент 3-го курса  группы КМБО-02-21 | Бредихин В.А. |
| Преподаватель | Крыжановский Ю.М. |
| Рецензент |  |
|  |  |

|  |  |  |
| --- | --- | --- |
| Работа представлена к защите | «\_\_»\_\_\_\_\_\_\_2023 г. |  |
|  |  |  |
| «Допущен к защите» | «\_\_»\_\_\_\_\_\_\_2023 г. |  |

Москва 2023

Содержание

[Задание №1 3](#__RefHeading___Toc7811_1268055582)

[Теоретическая часть 3](#__RefHeading___Toc7813_1268055582)

[Практическая часть 4](#__RefHeading___Toc7815_1268055582)

[Задание №2 5](#__RefHeading___Toc7817_1268055582)

[Теоретическая часть](#__RefHeading___Toc696_1272466978) 6

[Практическая часть 6](#__RefHeading___Toc698_1272466978)

[Задание №3 9](#__RefHeading___Toc2885_1914997794)

[Теоретическая часть](#__RefHeading___Toc2887_1914997794) 9

[Практическая часть 1](#__RefHeading___Toc2889_1914997794)0

[Приложения 14](#__RefHeading___Toc2891_1914997794)

**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=17.41, b=1.27, c=842.3, d=11.7, m=0.71

Теоретическая часть

Пусть A — точное число, a — приближенное.

Тогда число называется абсолютной погрешностью приближенного числа а.

Предельной абсолютной погрешностьючисла а называется любое число не меньшее

Относительной погрешностью приближенного числа а называется число

Предельной относительной погрешностьюприближённого числа а называется любое число не меньшее

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

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

Практическая часть

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

Следовательно, равенство b точнее

2)

Запишем *а* в верных знаках, для этого найдем абсолютную погрешность:

Округлим *а* до тысячных, получаем:

;

Округлим *а* до сотых, получаем:

Округлим  *а*  до десятых, получаем:

Округлим  *а*  до целого, получаем:

Следовательно, *а* записано в верных знаках.

Ответ: *а = 24.*

3)



Исходные данные:, a=17.41, b=1.27, c=842.3, d=11.7, m=0.71

Найдём абсолютные погрешности. Так как числа даны в верных знаках, то:

Вычислим выражение:

Вычислим производные:

Вычислим предельную погрешность выражения x:

Приведём число к верным знакам.

Округлим до тысячных:

Округлим до сотых:

Следовательно число x = 0.46 записано в верных знаках.

1. Задание №2
2. Найти границы действительных корней, используя схему Горнера. Отделить корни и уточнить каждый из них методом итерации с точностью до 0.001.
3. 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 коэффициентыито все действительные корни многочлена, если они есть, расположены не правее Для оценки нижней границы необходимо описанным выше способом найти верхнюю границу многочленапри этом нижняя граница многочленабудет 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 корняуравнения *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 приближенияи вычислением последовательностидо тех пор, покатак какПри этом задаваемый вышеуказанной формулой итерационный процесс будет сходиться тогда и только тогда, когда .

Практическая часть

1) Уточним корниспользуя схему Горнера.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 3 | 12 | 3 |
| 3 | 1 | 6 | 30 | 93 |

Корни *P(x):*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | -3 | 12 | -3 |
| 1 | 1 | -2 | 10 | 7 |

Отделим корни: P(3) = 93, P(-1) = -7*;*следовательно на данном отрезке один корень.

Уточним корень методом итерации. Преобразуем исходное уравнение:

на [-1, 3] возрастает на данном отрезке. Следовательно,

Оценим

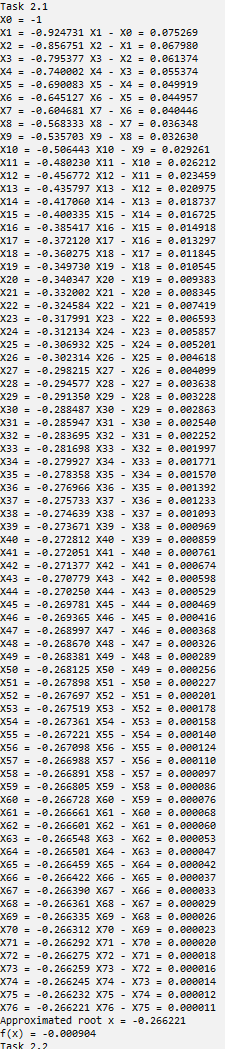
следовательно метод итерации сходится независимо от начального приближения.

Для расчета последовательностидо тех пор, пока при данных точности *ε,* функции *φ(x)* и начальном приближении была написана программа «L1». Программа написана на языке «C++» в операционной системе «Windows» с использованием IDE «Qt Creator». Для отладки проводились тесты программы на примерах с заранее известным ответом.

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

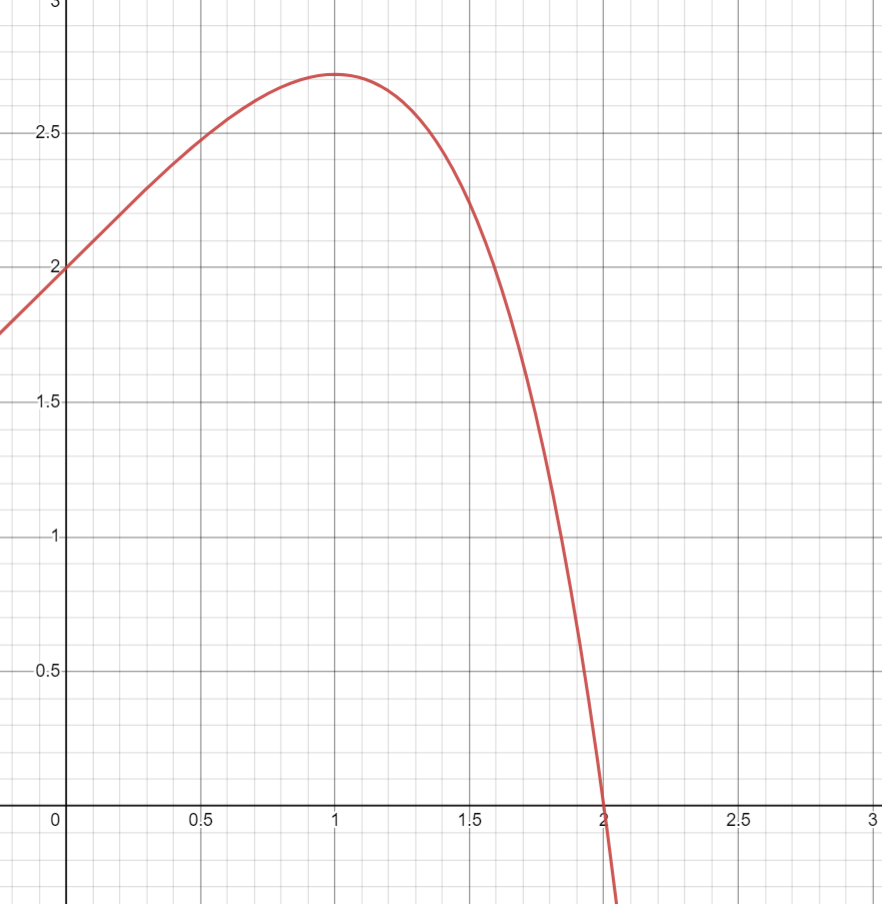
Результат: приближенный корень

Вывод программы:



2)

Графически отделим корни:



Уточним корень на отрезке [1.5; 2.5] методом итерации. Преобразуем исходное уравнение:

, где на отрезке [1.5; 2.5].

на отрезке [1.5; 2.5], следовательно убывает на этом отрезке, тогда:

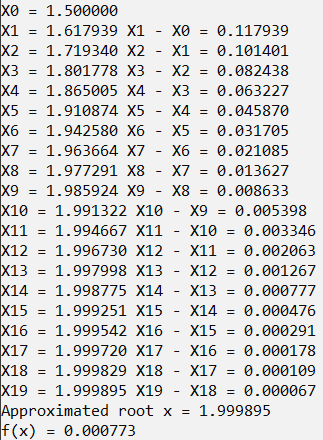
Оценим

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

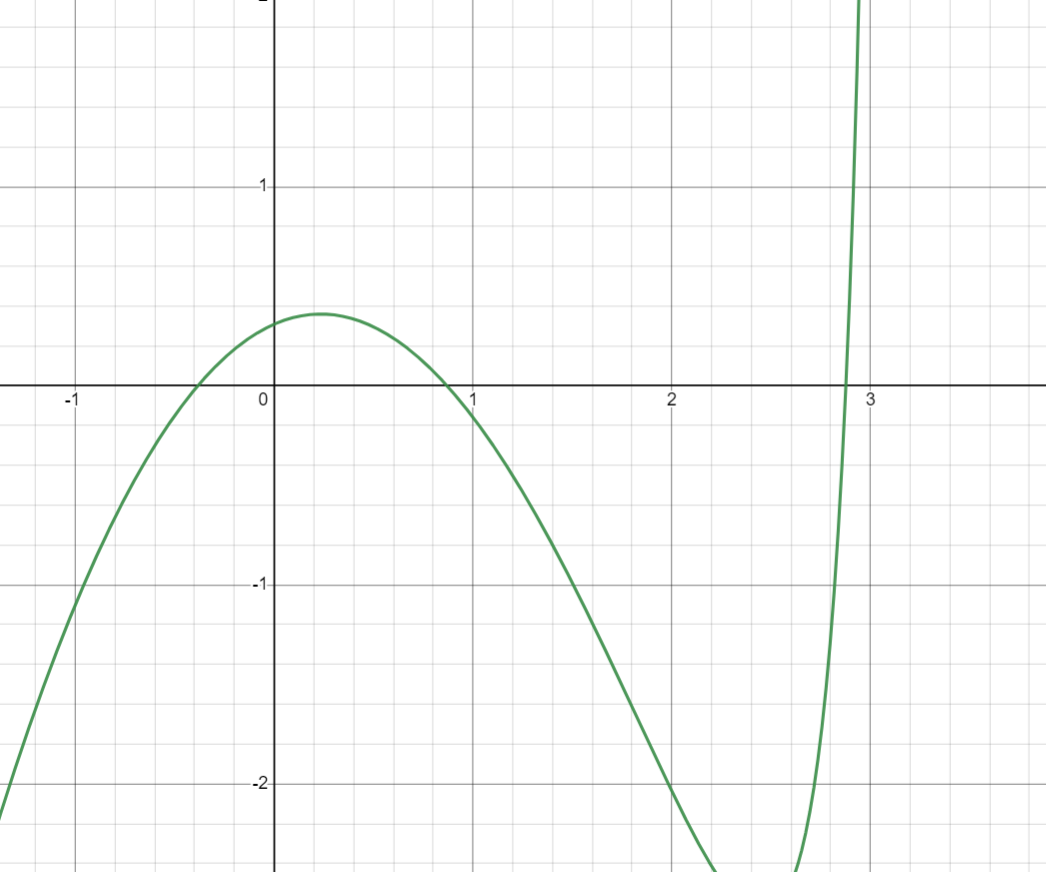
Для уточнения корняуравнения *f(x) =* 0, где функция *f(x)* имеет непрерывные и сохраняющие знак на данном отрезке первую и вторую производные,с заданной точностью *ε*, можно использовать метод хорд. Его суть заключается в построении последовательности приближенийгде - точка пересечения оси абсцисс и хорды между значениями функции в точках *c* иТочка *с —* неподвижный конец отрезка [a, b], в котором знак функции совпадает со знаком второй производной. В качестве начального приближениявыбирается тот конец отрезка [a, b], в котором знак функции несовподает 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 корняуравнения *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 приближенийМожно показать, что если функция на концах отрезка принимает разные знаки, аиотличны от нуля и сохраняют знаки, то при выборе начального приближенияпоследовательность 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. Условия сходимости совпадают с условиями сходимости для метода касательных.

Практическая часть

Графически отделим корни:

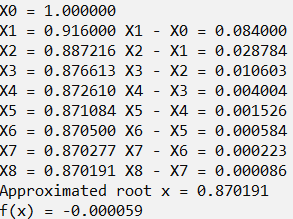


Уточним корень на отрезкеметодом хорд.

Рассчитаем последовательностьдо тех пор, пока не выполнится неравенство

Результат: приближенный корень

Вывод программы:



2)

Следовательно корень находится на отрезке [0.5, 1.5].

> 0 на отрезке [0.5, 1.5].

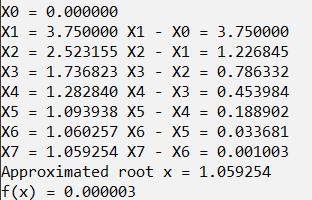
Поскольку обе производные сохраняют знак на [0.5, 1.5] мы можем уточнить корень используя метод касательных.

Возьмем начальное приближение

Рассчитаем последовательностьдо тех пор, пока не выполнится неравенство

Результат: приближенный корень

Вывод программы:



3)

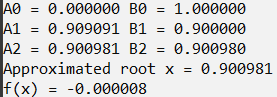
следовательно вещественная ось разбивается на два участка, где могут быть корни.

на отрезке [0,1] находится один корень, и на отрезке [10,12] находится корень так как

Уточним корень на отрезке [0, 1] комбинированным методом. Мы это можем сделать так как Рассчитаем последовательности до тех пор, пока не выполнится неравенство

Результат: приближенный корень

Вывод программы:



Приложения

**Исходный код**

**Lab1.cpp:**

#include <iostream>

#include <cmath>

*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 << fixed << "X" << i << " = " << nextX << " " << "X" << i << " - " << "X" << i - 1 << " = " << abs(nextX - prevX) << endl;

i++;

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

*return* nextX;

}

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

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

{

int i = 1;

cout << "A" << 0 << " = " << A0 << " "

<< "B" << 0 << " = " << B0 << endl;

*do*

{

*if* (A0 == B0)*return* A0;

A0 = chordsFunc(A0, B0);

B0 = tangentsFunc(B0);

cout << "A" << i << " = " << A0 << " "

<< "B" << i << " = " << B0 << endl;

i++;

} *while* (abs(f(abs(A0 + B0) / 2.)) > accuracy);

*return* A0;

}

double **f1**(double x) { *return* pow(x, 3) + 3 \* pow(x, 2) + 12 \* x + 3; }

double **iterFunc1**(double x) { *return* x - f1(x) / 93; }

double **f2**(double x) { *return* (2-x)\* exp(x);}

double **iterFunc2**(double x) { *return* x + f2(x) / 19; }

double **f3**(double x) { *return* tan(0.4 \* x + 0.3) - pow(x,2); }

double **chordsIterFunc3**(double x) { *return* x - f3(x) \* (x - 2) / (f3(x) - f3(2)); }

double **f4**(double x) { *return* pow(x, 3) - 0.1 \* pow(x, 2) + 0.4 \* x - 1.5; }

double **tangentsIterFunc4**(double x) { *return* x - f4(x) / (3 \* pow(x, 2) - 0.2 \* x + 0.4); }

double **f5**(double x) { *return* pow(x, 2) - 12\*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) / (2\*x-12); }

int **main**()

{

cout << "Task 2.1\n";

double e1 = iterMethod(-1, 0.001, iterFunc1, f1);

cout << "Approximated root x = " << e1 << endl;

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

cout << "Task 2.2\n";

double e2 = iterMethod(1.5, 0.001, iterFunc2, f2);

cout << "Approximated root x = " << e2 << endl;

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

cout << "Task 3.1\n";

double e3 = iterMethod(1, 0.0001, chordsIterFunc3, f3);

cout << std::fixed << "Approximated root x = " << e3 << endl;

cout << std::fixed << "f(x) = " << f3(e3) << endl;

cout << "Task 3.2\n";

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

cout << "Approximated root x = " << e4 << endl;

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

cout << "Task 3.3\n";

double e5 = combMethod(0, 1, 0.001, chordsIterFunc5, tangentsIterFunc5, f5);

cout << "Approximated root x = " << e5 << endl;

cout << fixed << "f(x) = " << f5(e5) << endl;

*return* 0;

}