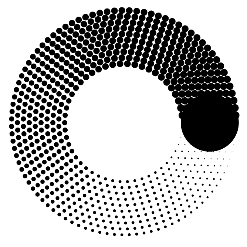
**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

***Факультет Информационных технологий***

***Кафедра Информатики и информационных технологий***

**направление подготовки**

**09.03.02 «Информационные системы и технологии»**

**ЛАБОРАТОРНАЯ РАБОТА № 3**

**Дисциплина:** Численные методы в компьютерных вычислениях

**Выполнил:** студент группы 231-338

Шаура Илья Максимович

**** (Фамилия И.О.)

**Дата, подпись** 29.03.2024  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Дата) (Подпись)

**Проверил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Фамилия И.О., степень, звание) (Оценка)

**Дата, подпись \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Дата) (Подпись)

Замечания: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Москва**

**2024**

**Вариант 14**

**Условия**

Найти корень уравнения для своего варианта.



Каждый студент пишет программу, и с ее помощью для своего варианта находит корень

При написании программы учесть ввод в интерактивном режиме следующих параметров: начальное приближение x0 (либо концы отрезка [a;b]) и точность ε. На выходе должны быть значение корня, точность ε и количество итераций. Сравнить скорость сходимости для разных методов.

Листинг 1 – Код консольной программы

|  |
| --- |
| #include <iostream>  #include <math.h>    int it\_count\_dich = 0;  int it\_count\_simple = 0;  int it\_count\_newton = 0;  int it\_count\_chord = 0;  double a, b, x0, eps;  double f(double x) {  return x\*x - 2 \* cos(x) + 1;  }  double f\_der(double x) {  return 2 \* x + 2 \* sin(x);  }  double f\_der2(double x) {  return 2 \* cos(x) + 2;  }  double phi(double x) {  return x - ((x\*x - 2 \* cos(x) + 1) / 4);  }  double simple\_it(double x0, double eps) {  double x\_curr = x0;  double x\_prev;  std::cout << "Enter \'x0\' (for simple it. method): ";  std::cin >> x0;  do {  it\_count\_simple++;  x\_prev = x\_curr;  x\_curr = phi(x\_prev);  } while (f(x\_curr - eps) \* f(x\_curr + eps) >= 0);  return x\_curr;  }  double dich\_it(double a, double b, double eps) {  double c;  while (f(a) \* f(b) > 0) {  std::cout << "Enter \'a\' (left): ";  std::cin >> a;  std::cout << "Enter \'b\' (right): ";  std::cin >> b;  if (f(a) \* f(b) > 0) std::cout << "There is no root between these values! Try again:\n";  }  do {  it\_count\_dich++;  c = (a + b) / 2.0;  if (f(a) \* f(c) < 0) b = c;  else if (f(b) \* f(c) < 0) a = c;  else {  std::cout << "The root is not found" << std::endl;  return 0;  }  } while (f(c - eps) \* f(c + eps) >= 0);  return c;  }  double newton\_it() {  it\_count\_newton = 0;  while (f(a) \* f(b) > 0) {  std::cout << "Enter \'a\' (left): ";  std::cin >> a;  std::cout << "Enter \'b\' (right): ";  std::cin >> b;  if (f(a) \* f(b) > 0) std::cout << "There is no root between these values! Try again:\n";    if (f(a) \* f\_der2(a) > 0) x0 = a;  else x0 = b;  }  double x1 = x0 - f(x0) / f\_der(x0);  while (f(x1 - eps) \* f(x1 + eps) >= 0) {  x0 = x1;  x1 = x0 - f(x0) / f\_der(x0);  it\_count\_newton++;  }  return x1;  }  double chord\_it() {  it\_count\_chord = 0;  double c;  while (f(a) \* f(b) > 0) {  std::cout << "Enter \'a\' (left): ";  std::cin >> a;  std::cout << "Enter \'b\' (right): ";  std::cin >> b;  if (f(a) \* f(b) > 0) std::cout << "There is no root between these values! Try again:\n";  }  do {  c = a - (b - a) \* f(a) / (f(b) - f(a));  if (f(c) \* f(a) < 0)  b = c;  else  a = c;  it\_count\_chord++;  } while (f(c + eps) \* f(c - eps) >= 0);  return c;  }  int main()  {  std::cout << "Enter \'epsilon\' (accuracy): ";  std::cin >> eps;    std::cout << std::endl;  std::cout << "----- NEWTON METHOD -----" << std::endl;  std::cout << "Root by Newton method: " << newton\_it() << std::endl;  std::cout << "Amount of iterations for Newton method = " << it\_count\_newton << std::endl;  std::cout << std::endl;  std::cout << std::endl;  std::cout << "----- CHORD METHOD -----" << std::endl;  std::cout << "Root by Newton method: " << chord\_it() << std::endl;  std::cout << "Amount of iterations for Newton method = " << it\_count\_chord << std::endl;  std::cout << std::endl;  std::cout << std::endl;  std::cout << "Epsilon (accuracy) = " << eps << std::endl;  } |

Сравнив количество итераций для данных методов, можно заключить, что метод дихотомии примерно в шесть раз быстрее метода простых итераций.