Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

**РЕШЕНИЕ УРАВНЕНИЙ С ОДНОЙ ПЕРЕМЕННОЙ**

**Отчет по лабораторной работе №1**

**По дисциплине**

**«Численные методы»**

Студент гр. 430-2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.А. Лузинсан

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Проверил: ст. преп. каф. АСУ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Е. Косова

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Томск 2022

**Оглавление**

[1 Цели и задачи 2](#_Toc98191545)

[2 Теория 3](#_Toc98191546)

[2.1 Интервальные методы 3](#_Toc98191547)

[2.2 Итерационные методы 3](#_Toc98191548)

[2.3 Комбинированный метод 5](#_Toc98191549)

[3 Ход выполнения лабораторной работы 6](#_Toc98191550)

[4 Пример решения 7](#_Toc98191551)

[Вывод 8](#_Toc98191552)

[Список использованных источников 9](#_Toc98191553)

[**Листинг программы** 10](#_Toc98191554)

# Цели и задачи

Изучить и применить на практике численные методы решения уравнений с одной переменной.

Необходимо реализовать три обязательных метода (дихотомии, хорд и Ньютона) и, по желанию, три дополнительных (комбинированный метод, метод итераций и золотого сечения).

# Теория

## Интервальные методы

Метод дихотомии заключается в следующем. Определяем половину отрезка и вычисляем f(c). Проверяем следующие условия:

1. Если |f(c)| < ε, то c — корень.

2. Если f (c)f (a) < 0, то корень лежит в интервале [a, c].

3. Если условие (2) не выполняется, то корень лежит на отрезке [c, b].

Продолжая процесс половинного деления в выбранных подынтервалах, можно дойти до сколь угодно малого отрезка, содержащего корень ξ.

Метод хорд можно рассматривать как одну из модификаций метода Ньютона.

В методе хорд интервал разбивается другой точкой:

В отличие от других интервальных методов, в методе хорд постоянное уменьшение длины интервала не гарантировано, поэтому погрешность рассчитывается по формуле итерационных методов:

В методе золотого сечения интервал разбивается двумя симметричными относительно границ интервала точками:

,

где .

## Итерационные методы

Методы Ньютона (касательных) и итераций являются итеративными (итерационными), на основе некоторого приближения корня *xk* они позволяют на каждой итерации получать новое приближение *xk+1*.

В итеративных методах оценивается расстояние между последним и предпоследним приближениями корня:

.

При этом нужно знать начальное приближение *x0*, а дальнейшие приближения на каждой *k+1*-й итерации находятся по итеративной формуле:

.

В методе Ньютона начальное приближение выбирается в соответствии со следующим условием: если в некоторой точке x произведение, то точка *x* является подходящей для начала итерационного процесса. Проверяются границы интервала:

На практике может наблюдаться ситуация, когда оба условия не выполняются. В этом случае вместо второго условия можно использовать оператор «иначе», либо воспользоваться вторым критерием.

Функция *φ(xk)* для метода Ньютона выглядит следующим образом:

.

В отличие от интервальных методов, длина исследуемого отрезка в которых на каждой итерации гарантированно уменьшается (например, для метода дихотомии – в два раза, для метода золотого сечения – в γ раз), в итеративных методах, в общем случае, расстояние между последовательными приближениями корня может иногда и увеличиваться. То же самое касается и значения функции в этих точках – оно может как уменьшаться, так и увеличиваться. Поэтому для некоторых функций условия могут не выполняться в течение довольно большого числа итераций (или вообще никогда). В этом случае итерации следует прекращать при выполнении хотя бы одного условия.

## Комбинированный метод

Комбинированный метод сочетает в себе сильные стороны методов хорд и Ньютона, и поэтому является достаточно эффективным для большого класса функций. Исключение интервалов выполняется по следующему алгоритму.

Сначала по формуле ищется точка пересечения хорды с осью x. Далее, если *f (ak) f "(ak) >0*, то точку *ak* можно переместить ближе к корню по формуле Ньютона. Тогда точка *bk* перемещается по формуле метода хорд. Данный факт изображён на рисунке 2.3.1.

Если же *f (bk) f "(bk) >0,* то, наоборот, точку *bk* можно переместить ближе к корню по формуле Ньютона, а точку *ak* – по формуле метода хорд, что показано на рисунке 2.3.1.

Два упомянутых условия достаточно проверять только один раз, если вторая производная не меняет своего знака на отрезке [a, b]. Но, т.к. это выполняется не для всех функций, лучше их проверять на каждой итерации.

Вместо второго условия можно использовать оператор «иначе», чтобы не возникла ситуация, когда оба условия не выполняются.

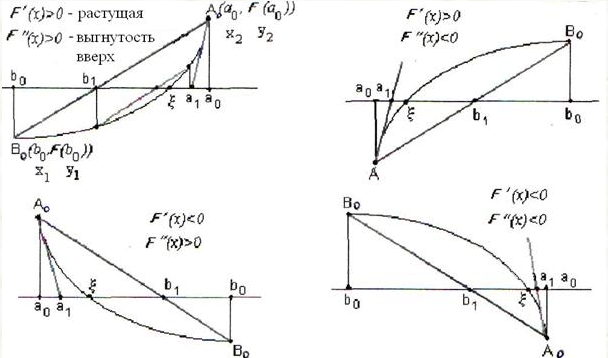


Рисунок 2.3.1 – Вариации использования комбинированного метода

# Ход выполнения лабораторной работы

Программа разделена на два основных класса: IInputDevice и EquationScalar. Первый класс отвечает за непосредственно считывание данных из файла или с клавиатуры, а также обеспечивает перенаправление потоков ввода данных для стандартного устройства ввода. Класс EqualationScalar обеспечивает вычисление корня функции по заданному методу. Если ввод изначально задавался с клавиатуры, то можно будет опробовать разные методы для одной и той же функции и посмотреть, какой из них будет эффективнее.

Класс содержит методы дихотомии, хорд, золотого сечения, комбинированный, Ньютона, итераций. Для определённого объекта можно вычислить корень на промежутке только с помощью одного метода, тем самым выставив поле с решением (\_res), полученную погрешность (\_eps\_new) и собственно выбранный метод (\_method).

Реализация каждого численного метода сделана в соответствии с теоретическим материалом методических указаний дисциплины.

При выводе данных в файл для x установлена точность по формуле:

.

Для fx также уставлена нотация вывода fixed – формат поля с плавающей запятой.

При выводе погрешности устанавливается научная нотация.

# Пример решения

Примеры работы методов на различных входных представлены в виде таблицы 4.1.

Таблица 4.1 – Примеры решений различных уравнений

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Метод | F(x) | a | b |  | x\* | F(x\*) |  |
| Дихотомии | - x + 3 \* x ^ 4 | 0.5 | 3 | 0.00001 | 0.69336 | -0.00001 | 9.53674e-06 |
| Хорд | 0.69336 | -0.00001 | 9.70976e-08 |
| Ньютона | 0.69336 | 0.00000 | 2.44129e-05 |
| Комбинированный | 0.69336 | 0.00000 | 1.71270e-09 |
| Итерационный | 0.69336 | -0.00000 | 2.44250e-07 |
| Золотого сечения | 0.69336 | -0.00000 | 1.75896e-06 |
| Дихотомии | x^2 - 16 | 2 | 8 | 0.0001 | 4 | -0.0002 | 9.1553e-05 |
| Хорд | 4 | -0.0000 | 1.0036e-05 |
| Ньютона | 4 | 0.0000 | 1.2193e-03 |
| Комбинированный | 4 | 0.0000 | 1.8584e-07 |
| Итерационный | 4.1 | 0.8100 | 9.0000e-01 |
| Золотого сечения | 4 | -0.0000 | 6.8305e-06 |

# Вывод

В ходе выполнения данной лабораторной работы я изучила теоретические сведения, связанные с численными методами решения уравнений с одной переменной.

Рассмотрела сильные и слабые стороны методов.

Например, недостатком метода Ньютона является необходимость вычисления значения производной *f ′(xn)* на каждой итерации. Соответственно, функция должна быть дифференцируема.

Метод дихотомии прост и надежен, но довольно медленный.

У метода хорд быстрота сходимости к решению сильно зависит от вида функции.

Комбинированный метод работает быстрее, чем методы хорд и касательных. Но, с не дифференцируемыми функциями также не работает.

# Список использованных источников

1. Вычислительные методы : учебное пособие / А.А.Мицель. — Томск : Эль Контент, 2013. — 198 с. (дата обращения: 01.03.2022).

**Листинг программы**

#pragma once

#include <iomanip>

#include <iostream>

#include "PolStr.h"

#define MAX\_ITER 100000

class EquationScalar

{

private:

double \_a, \_b, \_eps;

const char\* \_pstr;

char \_method;

double \_res;

double \_eps\_new;

std::string \_expr;

int \_count;

public:

EquationScalar(const char\* pstr, double a, double b, double eps, std::string expr, char method = '1') :

\_pstr{ pstr }, \_a(a), \_b(b), \_eps(eps), \_expr{ expr }, \_method(method), \_count(0), \_eps\_new(0), \_res(0)

{

//setResult(\_method);

}

std::string getType() const

{

switch (\_method)

{

case '1':

return "Метод Дихотомии";

case '2':

return "Метод Хорд";

case '3':

return "Метод Золотого Сечения";

case '4':

return "Комбинированный Метод";

case '5':

return "Метод Ньютона";

case '6':

return "Метод Итераций";

default:

return "Не Обнаружен";

}

}

int getCount()

{

return \_count;

}

friend std::ostream& operator<<(std::ostream& out, const EquationScalar& expr)

{

std::streamsize precision = std::cout.precision();

out << expr.getType() << ": \n"

<< expr.\_expr

<< "\nx' = " << std::setprecision(precision) << expr.\_res << "\n"

<< "f(x') = " << EvalPolStr(expr.\_pstr, expr.\_res, 0)

<< "\nВедённая погрешность: " << expr.\_eps

<< "\nПолученная погрешность вычислений: " << expr.\_eps\_new << std::endl;

return out;

}

void setResult(char choice)

{

\_method = choice;

switch (\_method)

{

case '1':

setDichotomyMethod();

break;

case '2':

setChordMethod();

break;

case '3':

setGoldenSectionMethod();

break;

case '4':

setCombinedMethod();

break;

case '5':

setNewtonMethod();

break;

case '6':

setIterationMethod();

break;

default:

throw std::invalid\_argument("Не найден подходящий метод решения");

}

}

//метод дихотомии - 1

double setDichotomyMethod()

{

double a = \_a, b = \_b, eps = \_eps, c;

double f\_a = EvalPolStr(\_pstr, a, 0), f\_b, f\_c;

int count = 0;

while ((b - a) / 2 >= eps && count < MAX\_ITER)

{

c = (a + b) / 2;

f\_c = EvalPolStr(\_pstr, c, 0);

if (f\_a \* f\_c <= 0)

{

b = c;

f\_b = f\_c;

}

else

{

a = c;

f\_a = f\_c;

}

count++;

}

\_count = count;

\_res = (a + b) / 2;

\_eps\_new = (b - a) / 2;

return \_res;

}

//метод хорд - 2

double setChordMethod()

{

double a = \_a, b = \_b, eps = \_eps;

double c;

double x; // начальное приближение

if (EvalPolStr(\_pstr, a, 0) \* EvalPolStr(\_pstr, a, 2) > 0)

x = a;

else x = b;

double f\_a = EvalPolStr(\_pstr, a, 0),

f\_b = EvalPolStr(\_pstr, b, 0),

f\_c;

int count = 0;

do {

c = a - f\_a \* (b - a)

/ (f\_b - f\_a);

f\_c = EvalPolStr(\_pstr, c, 0);

if (f\_a \* f\_c <= 0)

{

b = c;

f\_b = f\_c;

}

else

{

a = c;

f\_a = f\_c;

}

x = x - EvalPolStr(\_pstr, x, 0) / EvalPolStr(\_pstr, x, 1);

count++;

} while (abs(c - x) >= eps && abs(EvalPolStr(\_pstr, c, 0)) >= eps && count < MAX\_ITER);

\_count = count;

\_res = c;

\_eps\_new = abs(c - x);

return c;

}

//метод золотого сечения - 3

double setGoldenSectionMethod()

{

double a = \_a, b = \_b, eps = \_eps;

double c, d, h = (sqrt(5) + 1) / 2;

double f\_a = EvalPolStr(\_pstr, a, 0),

f\_d;

int count = 0;

while ((b - a) / 2 >= eps && abs(EvalPolStr(\_pstr, (a + b) / 2, 0)) >= eps && count < MAX\_ITER)

{

d = a + (b - a) / h; // новая правая граница

c = a + b - d; // c - a = b - d / или новая левая граница

f\_d = EvalPolStr(\_pstr, d, 0);

if (f\_a \* f\_d <= 0)

b = d;

else

{

a = c;

f\_a = EvalPolStr(\_pstr, c, 0);

}

count++;

}

\_count = count;

\_res = (a + b) / 2;

\_eps\_new = (b - a) / 2;

return \_res;

}

//комбинированный метод - 4

double setCombinedMethod()

{

double a = \_a, b = \_b, eps = \_eps;

double f\_a0 = EvalPolStr(\_pstr, a, 0);

double c;

int count = 0;

do {

// вычисляем приближение

c = a - f\_a0 \* (b - a)

/ (EvalPolStr(\_pstr, b, 0) - f\_a0);

// если f(a)f''(a) > 0

if (f\_a0 \* EvalPolStr(\_pstr, a, 2) > 0)

{

// точка а перемещается по методу Ньютона(касательных)

a = a - f\_a0 / EvalPolStr(\_pstr, a, 1);

f\_a0 = EvalPolStr(\_pstr, a, 0);

// а точка b перемещается по методу хорд

b = c;

}

else// т.е. если f(b)f''(b) > 0

{

// точка а перемещается по методу хорд

a = c;

// а точка b перемещается по методу Ньютона

b -= EvalPolStr(\_pstr, b, 0) / EvalPolStr(\_pstr, b, 1);

}

count++;

} while (/\*(b - a) / 2 > eps &&\*/ abs(EvalPolStr(\_pstr, (a + b) / 2, 0)) >= eps && count < MAX\_ITER);

\_count = count;

\_res = (a + b) / 2;

\_eps\_new = (b - a) / 2;

return \_res;

}

//метод Ньютона - 5

double setNewtonMethod()

{

double a = \_a, b = \_b, eps = \_eps;

double x0; // начальное приближение

if (EvalPolStr(\_pstr, a, 0) \* EvalPolStr(\_pstr, a, 2) > 0)

x0 = a;

else x0 = b;

double x1 = x0;

int count = 0;

do {

x1 = x1 - EvalPolStr(\_pstr, x1, 0) / EvalPolStr(\_pstr, x1, 1);

if (abs(x1 - x0) < eps) break;

x0 = x1;

count++;

} while (count < MAX\_ITER);

\_count = count;

\_res = x1;

\_eps\_new = abs(x1 - x0);

return x1;

}

double maxf(double a, double b)

{

const double goldenRatio = (1 + sqrt(5)) / 2; // "Золотое" число

double x1, x2; // Точки, делящие текущий отрезок в отношении золотого сечения

while (fabs(b - a) >= \_eps)

{

x1 = b - (b - a) / goldenRatio;

x2 = a + (b - a) / goldenRatio;

if (EvalPolStr(\_pstr, x1, 1) <= EvalPolStr(\_pstr, x2, 1))

a = x1;

else

b = x2;

}

return (a+b)/2;

}

//метод итераций - 6

double setIterationMethod()

{

double a = \_a, b = \_b, eps = \_eps;

double x0 = a, x1 = b;

double f\_x0 = EvalPolStr(\_pstr, x0, 1), f\_x1 = EvalPolStr(\_pstr, x1, 1);

double max = maxf(a, b);

int count = 0;

do {

x1 = x1 - EvalPolStr(\_pstr, x1, 0) / max;

if (abs(x0 - x1) <= eps) break;

x0 = x1;

count++;

} while (count < MAX\_ITER);

\_count = count;

\_eps\_new = abs(x0 - x1);

\_res = x0 - EvalPolStr(\_pstr, x0, 0) / max;

return \_res;

}

};