Министерство образования и науки Российской Федерации

Санкт-Петербургский политехнический университет Петра Великого

Институт прикладной математики и информатики

Лабораторная работа по дисциплине

Численные методы

«Решение нелинейных уравнений»

Выполнил

студент гр.5030102/20001 Дрекалов Н.С.

Преподаватель Добрецова С.Б.

Санкт-Петербург

2023

Оглавление

[**Формулировка задачи** 3](#_Toc145969534)

[**Теорема о верхней границе** 3](#_Toc145969535)

[**Выделение корня** 4](#_Toc145969536)

[**Метод половинного деления** 4](#_Toc145969537)

[**Метод простых итераций** 5](#_Toc145969538)

[**Тестовый пример к методам** 5](#_Toc145969539)

[**Контрольные тесты** 7](#_Toc145969540)

[**Модульная структура программы** 8](#_Toc145969541)

[**Численный анализ методов** 9](#_Toc145969542)

[Вывод 11](#_Toc145969543)

## **Формулировка задачи**

Найти решение нелинейного уравнения с помощью метода половинного деления и метода простых итераций.

*Функции:*

## **Теорема о верхней границе**

*Условия применимости:*

* Функция f(x) – многочлен.

*Алгоритм метода:*

* Посчитать границу по формуле , где m ­ номер первого отрицательного коэффициента в ряду и a′ ­ наибольший по модулю отрицательный коэффициент. Проделать то же самое для -x, 1/x, -1/x. Таким образом пара значений (x, 1/x) дает диапазон положительных значений корня, а пара (-x, -1/x) – отрицательных.

*Полученные отрезки:*

* “x”: ; m = 4, a’ = -1, a0 = 2;
* “-x”: ; m = 2, a’ = -8, a0 = 2;
* “1/x”: ; m = 2, a’ = -8, a0 = 1;
* “-1/x”: ; m = 2, a’ = -8, a0 = 1;

Таким образом, положительные корни находятся в интервале (0.26; 1.85), а отрицательные – (-3, -0.26)

## **Выделение корня**

*Найдем производную полинома:*

Производная положительна для любого x > 0, в частности, для x из интервала (0.26; 1.85), то есть на данном интервале один корень.

*Найдем производную трансцендентной функции:*

Функция всюду непрерывна и при этом f(-6) > 0, f(-4) < 0, и на этом участке производная знакопостоянная. То есть, на интервале (-6; -4) есть корень и он единственный.

## **Метод половинного деления**

*Условия применимости:*

* Если функция f(x) непрерывна на отрезке [a,b] и на его концах принимает разные знаки, то методом половинного деления можно найти корень уравнения.

*Алгоритм метода:*

Допустим, что единственный корень уравнения лежит на отрезке [a,b].

a0 = a

b0 = b

1. Расчет середины отрезка [а0,b0].

c0 = (a0 + b0)/2

1. k-итерация (k=1,2,3…). Сужение отрезка [аk-1,bk-1].

Если f(аk-1)\*f(c k-1)<0, то аk = а k-1, bk **=** c k-1

Если f(а k-1)\*f(c k-1)>0, то аk = c k-1, bk **=** b k-1

1. Расчет середины отрезка [аk,bk].

ck = (ak + bk)/2

1. Проверка условия итерационного процесса. Если условие выполнено, то сk – корень уравнения:

| сk - сk-1 | < ℇ

Иначе перейти к 2 пункту.

**Метод простых итераций**

*Условия применимости:*

* ∃q : |ϕ ′ (x)|≤ q < 1 для ∀x ∈ [a, b]
* Корень x∗ ∈ [α, β] ⊂ [a, b], где ,

*Алгоритм метода:*

1. Находим M и m – границы производной f’ на отрезке [a; b].
2. Находим и , x1 = a, x2 = b.
3. Проверка условия остановки итерационного процесса, если условие выполнено, то – корень уравнения:

Иначе перейти к 4 пункту.

1. Получаем новую пару так, что , а .
2. Переходим к пункту 3.

**Тестовый пример к методам**

1. Решим с помощью метода половинного деления уравнение x^2 – 2\*x – 3= 0

Корнями этого уравнения являются значения -1 и 3. Найдем второй корень с помощью метода половинного деления на промежутке [1,4].

Проверим лежит ли в этом промежутке наш корень, т.е. функция принимает значение разных знаков на концах промежутков.

Формула для проверки условия: f(а)\*f(b) < 0

a0=1

b0=4

f(1) = 1 – 2 – 3 = -4 < 0

f(4) = 16 – 8 – 3 = 5 > 0

Функция принимает значение разных знаков на концах промежутков. Следовательно, мы можем продолжить решение.

x0 = (a0 + b0)/2 = (1+4)/2 = 2.5

Получилось два промежутка. Из них надо выбрать тот на котором функция принимает значение разных знаков.

f(2.5) = 6.25 – 5 – 3 = -1.75 < 0

Следовательно, промежуток [2.5, 4] нам подходит т.к f(2.5)\*f(4) < 0.

a1=2.5

b1=4

x1 = (a1 + b1)/2 = (2.5+4)/2 = 3.25

Получилось два промежутка. Из них надо выбрать тот на котором функция принимает значение разных знаков.

f(3.25) = 10.5625 - 6.5 – 3 = 1.0625 > 0

Следовательно, промежуток [2.5, 3.25] нам подходит т.к f(2.5)\*f(3.25) < 0.

a2=2.5

b2=3.25

x2 = (a2 + b2)/2 = (2.5+3.25)/2 = 2.875

Получилось два промежутка. Из них надо выбрать тот на котором функция принимает значение разных знаков.

f(2.875) = 8.265625 – 5.75 – 3 = -0.484375 < 0

Следовательно, промежуток [2.875, 3.25] нам подходит т.к f(2.875)\*f(3.25) < 0.

a3=2.875

b3=3.25

x3 = (a3 + b3)/2 = (2.875+3.25)/2 = 3.0625

По итогу мы все ближе приближаемся к корню x=3.

1. Решим это же уравнение модифицированным методом Ньютона.

Так же найдем второй корень (x=3) с помощью модифицированного метода Ньютона.

x0 = 4

Проверим, что первая и вторая производная неравна нулю:

f ’(x0) = 2\*x – 2 = 6 != 0

f ’ ’(x0) = 2 != 0

Проверим последнее условие сходимости (f(x0)\* f ’’(x0)>0):

f(x0)\* f ’’(x0) = (16 – 8 – 3)\*2 = 10 > 0

Следовательно модифицированный метод Ньютона сходится к точному решению.

Зададим константу:

const= f ’(x0) = 2 \* 4 – 2 = 6

Проводим расчеты, для получения корня:

x1 = x0 – f(x0)/const = 4 – (16 – 8 – 3)/6 = 19/6 ~ 3.16

Продолжаем алгоритм:

x2 = x1 – f(x1)/const = 19/6 – (361/36 – 19/3 – 3)/6 = 659/216 ~ 3.05

По итогу мы все ближе приближаемся к корню x=3.

**Контрольные тесты**

Метод половинного деления:

1. y(x) = 3\*x^4 + 4\*x^3 – 12\*x^2 + 1; x∈[-3, 2.5]; ℇ = 0.1; x = -2.8125
2. y(x) = 3\*x^4 + 4\*x^3 – 12\*x^2 + 1; x∈[-3, 2.5]; ℇ = 0.001; x = -2.764648
3. y(x) = 5^x - 6\*x – 3; x∈[-1, 0]; ℇ = 0.001; x = -0.415039
4. y(x) = 5^x - 6\*x – 3; x∈[-1, 0]; ℇ = 0.00001; x = -0.414467

Модифицированный метод Ньютона

1. y(x) = 3\*x^4 + 4\*x^3 – 12\*x^2 + 1; x∈[-3, 2.5]; ℇ = 0.01; x = -2.7686

const = -52.5

1. y(x) = 3\*x^4 + 4\*x^3 – 12\*x^2 + 1; x∈[-3, 2.5]; ℇ = 0.001; x = -2.764856

const = -52.5

1. y(x) = 5^x - 6\*x – 3; x∈[-1, 0]; ℇ = 0.001; x = -0.414506

const = -4.390562

1. y(x) = 5^x - 6\*x – 3; x∈[-1, 0]; ℇ = 0.00001; x = -0.414465

const = -4.390562

**Модульная структура программы**

/// Класс, который имеет два поля - для хранения числа итераций и для хранения корня

class Result {

public:

explicit Result(unsigned iterNumber, double root) : iterNumber(iterNumber), root(root) {};

const unsigned iterNumber = 0;

const double root;

};

#define MAX\_ITER\_NUM ULLONG\_MAX

/\*\*

\* **\brief** Функция находит корень в заданном промежутке методом ПД с точностью эпсилон

\* **@param** f функция, с у которой ищется корень

\* **@param** a левая граница

\* **@param** b правая граница

\* **@param** epsilon точность

\* **@return** тип Result в котором содержатся два поля - корень и число итераций

\*/

Result findRoot(double(&f)(double), double a, double b, double epsilon);

/\*\*

\* **\brief** Функция находит корень в заданном промежутке методом простых итераций с точностью эпсилон для полинома

\* **@param** f полином

\* **@param** epsilon точность

\* **@return** тип Result в котором содержатся два поля - корень и число итераций

\*/

Result findRootForPolynom(double(&f)(double), double epsilon);

/\*\*

\* **\brief** Функция находит корень в заданном промежутке методом простых итераций с точностью эпсилон для трансцендентой функции

\* **@param** f трансцендентная функция

\* **@param** epsilon точность

\* **@return** тип Result в котором содержатся два поля - корень и число итераций

\*/

Result findRootForComplex(double(&f)(double), double epsilon);

/\*\*

\* **\brief** функция, считающая значения полинома в точке x

\* **\param** x переменная

\* **\return** значение полинома в точке x

\*/

double polyVar11(const double x);

/\*\*

\* **\brief** функция, считающая значения транс фукнции в точке x

\* **\param** x переменная

\* **\return** значение функции в точке x

\*/

double complexVar11(const double x);

/\*\*

\* **\brief** функция, считающая корень для полинома для различной точности и выводящая всё на экран

\*/

void testFirstFunc();

/\*\*

\* **\brief** функция, считающая корень трансцедентной фукнции для различной точности и выводящая всё на экран

\*/

void testSecondFunc();

## **Численный анализ методов**

Рассмотрим зависимости количества итераций от точности приближения.



На графике мы наблюдаем, что метод половинного деления выполняет меньшие количество итераций, чем модифицированный метод Ньютона, а следовательно работает быстрее. Также метод половинного деления работает медленнее, чем функция fzero.

Рассмотрим зависимость ln разности с истинным корнем от точности



Вывод

Можно сделать вывод о том, что при работе с полином лучше применять метод половинного деления, чем модифицированный метод Ньютон. Так как происходят меньше итераций, а следовательно тратится меньше времени на нахождение корня. При работе с трансцендентным уравнениями, наоборот, лучше применять модифицированный метод Ньютона.