

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Кафедра Системотехніки

Звіт  
З лабораторної роботи №2  
на тему "Чисельні методи розв'язання алгебраїчних та трансцендентних  
рівнянь"

Виконав:  
ст. гр. АКТАКІТ-20-3  
Бобков М. В.

Перевірив:  
доц. каф. системотехніки  
Ситнікова П. Е.

Харків 2021

2.1 Цель работы: изучение приближенных методов решения алгебраических и трансцендентных уравнений.

## 2.2 Ход работы

Задание:

Выполнить процедуру отделения корней и запрограммировать алгоритмы поиска корней уравнения для заданных методов. Расчёты сделаны на примере второго варианта. Значение приведены в табл.2.1.

Таблица 2.1 – Данные для расчётов

Вариант	Уравнение	Методы	Точность
2	$x^3 - 4x^2 + x + 6$	Касательных Комбинированный Половинного деления	0,001

### 2.2.1 Отделение корней

1) Область определения функции  $x \in (-\infty; +\infty)$

2) Найдём производную функции  $f'(x) = 3x^2 - 8x + 1 \rightarrow x_1 = 0,13; x_2 = 2,5$ .

3) Определим интервалы монотонности функции  $(-\infty; 0,13]$ ,  $[0,13; 2,5]$ ,  $[2,5; +\infty)$ .

4) Функция принимает противоположные знаки в граничных точках каждого интервала. Значит, внутри интервалов содержится по одному корню данного уравнения.

График функции приведен на рис. 2.1.

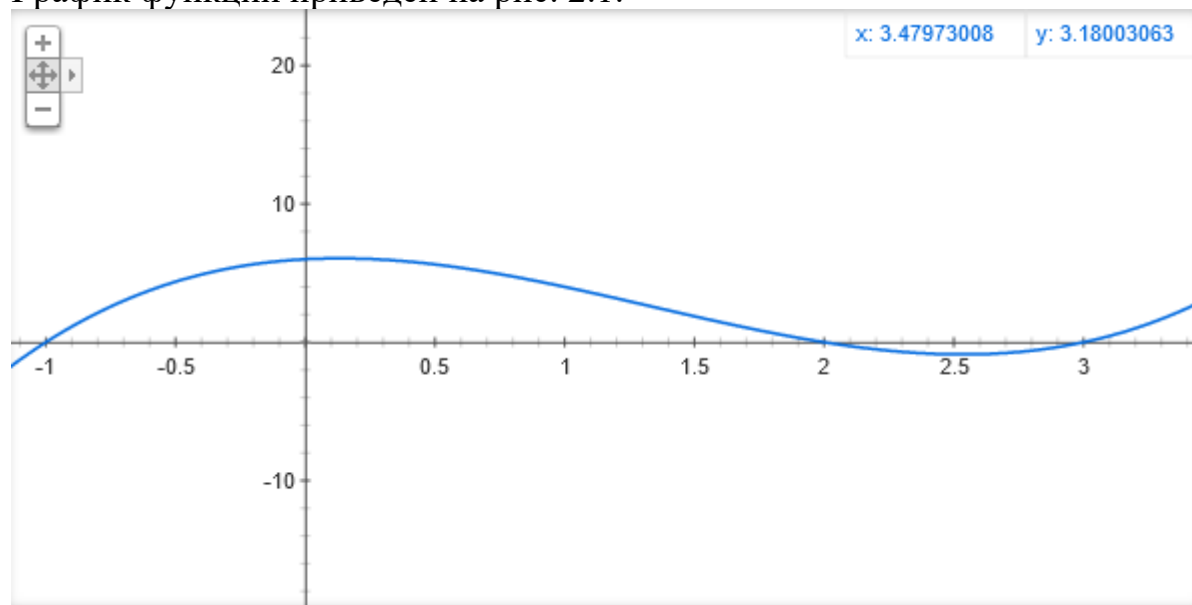


Рисунок 2.1 – График исследуемой функции

### 2.2.2 Уточнение корней

### 2.2.2.1 Метод половинного деления

Процесс уточнения корня заключается в построении последовательности вложенных друг в друга отрезков каждый из которых содержит корень уравнения. Процесс деления отрезка пополам следует проводить до тех пор, пока не будет выполнено неравенство  $(b_k - a_k) \leq 2\varepsilon$ . В качестве приближенного значения корня принимается середина  $k$ -го отрезка локализации.

Код программы приведен ниже. Написано на языке программирования C.

```
#include "main.h"

#define ACCURACY 0.001

float solve_function( float variable ) {
    return (powf(variable, 3) - 4 * powf(variable, 2) + variable + 6);
}

bool verify_data(float left_border, float right_border) {
    bool res = ((right_border - left_border) < (2 * ACCURACY));
    return res;
}

bool verefing(float left_border, float right_border)
{
    bool res = false;
    if (verify_data(left_border, right_border)) {
        printf("\n\rAnswer is: %f", ((left_border + right_border)/2) );
        res = true;
    }
    return res;
}

void half_deviding_method(float range_left_border, float range_right_border )
{
    float avarage = 0;
    float function_res = 0;
    float condition = 0;
    int iteration_count=1;
    while (1){
        avarage = ((range_left_border + range_right_border) / 2);
        function_res = solve_function(avarage);
        if (function_res == 0){
            printf("Answer is: %f\n", avarage); break;
        }
        condition = solve_function(range_left_border) *
solve_function(avarage);
        if (condition < 0){
            range_right_border = avarage;
            if (verefing(range_left_border, range_right_border)) {
                printf("\t Iteration #%d", iteration_count);
                break;
            }
        }
        condition = solve_function(avarage) *
solve_function(range_right_border);
        if (condition < 0){
            range_left_border = avarage;
        }
    }
}
```

```

        if (verefing(range_left_border, range_right_border)) {
            printf("\t Iteration #%d", iteration_count);
            break;
        }
    }
    iteration_count++;
}
}

int main() {

    float range_left_border = 1.3;
    float range_right_border = 2.5;

    half_deviding_method(-2, range_left_border);
    half_deviding_method(range_left_border, range_right_border);
    half_deviding_method(range_right_border, 4);

    return 0;
}

```

Вывод в консоли:

```

Answer is: -1.000171      Iteration #11
Answer is: 2.000195      Iteration #10
Answer is: 3.000244      Iteration #10
E:\Education\ЧМ\ChM\Debug\ChM.exe (процесс 14104) завершил работу с кодом 0.

```

### 2.2.2.2 Метод касательных(Ньютона)

Идея метода Ньютона заключается в том, что на достаточно малом отрезке  $[a;b]$  дуга графика функции  $y=f(x)$  заменяется касательной. В качестве приближенного значения корня  $x$  принимается точка пересечения касательной с осью абсцисс.

Код программы приведен ниже. Написано на языке программирования C.

```

#include "main.h"

#define ACCURACY 0.001

float solve_function( float variable ) {
    return (powf(variable, 3) - 4 * powf(variable, 2) + variable + 6);
}

float solve_derivate(float variable) {
    return (3 * powf(variable, 2) - 8 * variable + 1);
}

float calc_border_newton(float value) {
    return value - (solve_function(value) / solve_derivate(value));
}

void newton_method(float range_left_border, float range_right_border) {
    float new_range_right_border, new_range_left_border;
    float function_double_derivative_res = 0.4;
    float condition = 0;
    int iteration_count = 1;
    while (1) {
        condition = solve_double_derivative((range_left_border +
range_right_border) / 2) * solve_function(range_left_border);
        if (condition > 0) {

```

```

        new_range_left_border = calc_border_newton(range_left_border);
        if ((fabsf(new_range_left_border - range_left_border)) <
ACCURACY) {
            printf("\n\rAnswer is: %f", new_range_left_border);
            printf("\t Iteration #%d", iteration_count);
            break;
        }
        else {
            range_left_border = new_range_left_border;
        }
    }
    condition = solve_double_derivate((range_left_border +
range_right_border) / 2) * solve_function(range_right_border);
    if (condition > 0) {
        new_range_right_border = calc_border_newton(range_right_border);
        if ((fabsf(new_range_right_border - range_right_border)) <
ACCURACY) {
            printf("\n\rAnswer is: %f", new_range_right_border);
            printf("\t Iteration #%d", iteration_count);
            break;
        }
        else {
            range_right_border = new_range_right_border;
        }
    }
    iteration_count++;
}
}
int main() {

    float range_left_border = 1.3;
    float range_right_border = 2.5;

    newton_method(-2, range_left_border);
    newton_method(range_left_border, range_right_border);
    newton_method(range_right_border, 4);

    return 0;
}

```

Вывод в консоли:

```

Answer is: -1.000000      Iteration #5
Answer is: 2.000000      Iteration #4
Answer is: 3.000000      Iteration #5
E:\Education\ЧМ\ЧМ\Debug\ЧМ.exe (процесс 13892) завершил работу с кодом 0.

```

### 2.2.2.3 Комбинированный метод

Идея комбинированного метода заключается в том, что с одной стороны отрезка локализации  $[a;b]$  применяется метод хорд, с другой – метод касательных.

Код программы приведен ниже. Написано на языке программирования C.

```

#include "main.h"

#define ACCURACY 0.001
float solve_function( float variable ) {
    return (powf(variable, 3) - 4 * powf(variable, 2) + variable + 6);
}

```

```

}
float solve_derivate(float variable) {
    return (3 * powf(variable,2) - 8 * variable + 1);
}

float solve_double_derivate(float variable) {
    return (6* variable - 8);
}

void calc_borders(float left_in, float right_in, bool condition, float*
left_out, float * right_out) {

    *left_out = (condition? left_in: right_in) -
                (solve_function((condition ? left_in : right_in)) /
(solve_function((condition ? right_in : left_in)) -
                solve_function((condition ? left_in : right_in))) *
((condition ? right_in : left_in) - (condition ? left_in : right_in)));
    *right_out = (condition ? right_in : left_in) -
(solve_function((condition ? right_in : left_in)) /
solve_derivate(condition?right_in : left_in));
}

void combined_method(float range_left_border, float range_right_border) {
    float condition = 0;
    int iteration_count = 1;
    while (1) {
        // condition = solve_double_derivate((range_left_border +
range_right_border) / 2) * solve_function(range_left_border);
        condition = solve_double_derivate((range_left_border +
range_right_border) / 2) * solve_derivate((range_left_border +
range_right_border) / 2);
        if (condition > 0) {
            calc_borders(range_left_border, range_right_border, true,
&range_left_border, &range_right_border);
            if (fabsf(range_right_border - range_left_border) < ACCURACY) {
                printf("\r\nAnswer is: %f", (range_right_border +
range_left_border) / 2);
                printf("\t Iteration #%d", iteration_count);
                break;
            }
        }
        // condition = solve_double_derivate((range_left_border +
range_right_border) / 2) * solve_function(range_right_border);
        if (condition < 0) {
            calc_borders(range_left_border, range_right_border, false,
&range_left_border, &range_right_border);
            if (fabsf(range_right_border - range_left_border) < ACCURACY) {
                printf("\r\nAnswer is: %f", (range_right_border +
range_left_border) / 2);
                printf("\t Iteration #%d", iteration_count);
                break;
            }
        }
        iteration_count++;
    }
}

int main() {

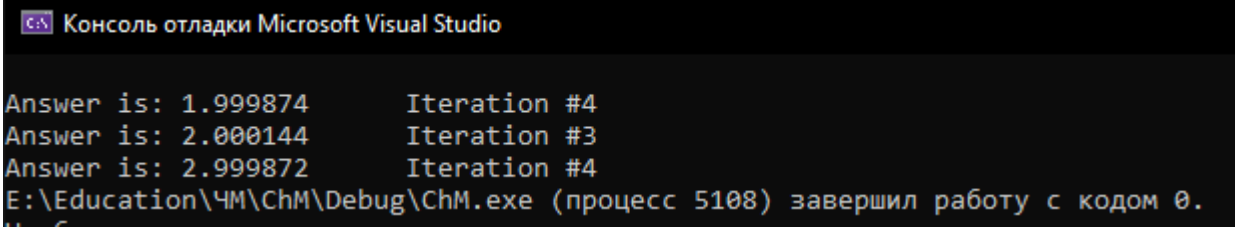
    float range_left_border = 1.3;
    float range_right_border = 2.5;

    combined_method(-5,range_left_border);
    combined_method(range_left_border,range_right_border);
    combined_method(range_right_border,4);
}

```

```
}  
    return 0;  
}
```

Вывод в консоли:



Консоль отладки Microsoft Visual Studio

```
Answer is: 1.999874      Iteration #4  
Answer is: 2.000144      Iteration #3  
Answer is: 2.999872      Iteration #4  
E:\Education\ЧМ\ChM\Debug\ChM.exe (процесс 5108) завершил работу с кодом 0.
```

## ВЫВОДЫ

Во время работы ознакомились с разными методами решения систем алгебраических и трансцендентных уравнений, разобрали на практике многие из них. Проверили их принцип работы на поставленных задачах, значения переменных для которых приведены по-вариантно.