Міністерство освіти, науки, молоді та спорту України
Національний технічний університет України
«Київський політехнічний інститут»
Фізико-технічний інститут

Лабораторна робота №1

Методи обчислень
«Розв’язання нелінійних рівнянь»

Варіант 4

Підготував:
студент 3 курсу
групи ФІ-84
Коломієць Андрій Юрійович

Викладач:
Стьопочкіна Ірина Валеріївна

Київ – 2021

Допрограмний етап

Варіант №4

$$f(x) = -x^4 + 3x^3 - 2x + 4.$$

Завдання допрограмного етапу:

1. Визначити кількість дійсних коренів рівняння (теорема Гюа, Штурмана)
2. Відокремити дійсні корені рівняння (ТВН)
3. До аналізу комплексних коренів застосувати теорему про кільце

Результат: послідовність проміжків, кожен з яких містить лише один дійсний корінь рівняння

Розв'язок

1. За теоремою Гюа потрібно визначити чи є комплексно спряжені корені:

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_0 = 0$$

Якщо $\exists k: 0 < k < n$, $a_k^2 < a_{k-1} a_{k+1}$ то рівняння має хоча б одну пару комплексно спряжених коренів

Перевірка:

if $a_0 = 4$ —

if $a_1 = -2$ тоді $(-2)^2 < 0 \cdot 4 \Rightarrow 4 < 0$ - немає

if $a_2 = 0$ тоді $0^2 < -2 \cdot 3 \Rightarrow 0 < -6$ - немає

if $a_3 = 3$ тоді $3^2 < -1 \cdot 0 \Rightarrow 9 < 0$ - немає

if $a_4 = -1$ —

Достатня умова існування комплексних коренів не виконується

Але якщо теорема Гюа не показала наявності співвідношення то це не означає, що комплексно спряжених коренів взагалі не буде

За теоремою трохи нище можна визначити межі в котрих будуть лежати всі корені:

1) $A = \max\{|a_{n-1}|, |a_{n-2}|, \dots, |a_0|\}$, $B = \max\{|a_n|, |a_{n-1}|, \dots, |a_1|\}$

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_0 = 0.$$

a_i – коеф. поліп. $P_n(x)$

Тоді модулі всіх коренів x_k, $k = 1, \dots, n$ рівняння $P_n(x) = 0$ лежать в кільці:

$$r = \frac{|a_0|}{|a_0| + B} < |x_k| < \frac{|a_n| + A}{|a_n|} = R$$

де r та R – нижня та верхня границі додатніх коренів

Перевірка:

$A = \max\{|3|, |0|, |-2|, |4|\} = 4$

$B = \max\{|-1|, |3|, |0|, |-2|\} = 3$

Межі в котрих лежать всі розв'язки:

$$\frac{4}{4+3} < |x_k| < \frac{1+4}{1}$$

$$4/7 < |x_k| < 5.$$

Знайдемо R^+, R_+, R^-, R_- відповідно верхню, нижню границі додатніх коренів і верхню, нижню від'ємних за теоремою про верхню межу:

① $f(x) = -x^4 + 3x^3 - 2x + 4 = 0 \Rightarrow x^4 - 3x^3 + 2x - 4 = 0 \Rightarrow$

$\Rightarrow$ $n = 4$, $m = 3$, $B_1 = 4$ $\Rightarrow$ $R = 1 + 4 = 5$. $\leftarrow$ Верхня межа для додатніх коренів

② $f_1(x) = x^4 f\left(\frac{1}{x}\right) = x^4\left(-\frac{1}{x^4} + 3\frac{1}{x^3} - 2\frac{1}{x} + 4\right) = 0 \Rightarrow$

$\Rightarrow 4x^4 - 2x^3 + 3x - 1 = 0 \Rightarrow$ $n = 4$, $m = 3$ та $B_1 = 2 \Rightarrow R_1 = 1 + 1/2 = 3/2$

$1/R_1$ – це нижня межа додатніх коренів, тобто $\frac{1}{R_1} = \frac{1}{3/2} = 2/3$.

③ $f_2(x) = f(-x) = -x^4 - 3x^3 + 2x + 4 = 0 \Rightarrow$

$\Rightarrow x^4 + 3x^3 - 2x - 4 = 0 \Rightarrow$ $n=4$, $m=1$, $B_1=4$ $\Rightarrow R_2 = 1 + \sqrt[3]{4} = 2{,}587$

$-R_2$ – це нижня границя для відємних коренів, тобто $-R_2 = -2{,}587$

④ $f_3(x) = x^4 f(-1/x) = x^4\left(-\frac{1}{x^4} - 3\frac{1}{x^3} + 2\cdot\frac{1}{x} + 4\right) = 0 \Rightarrow$

$\Rightarrow 4x^4 + 2x^3 - 3x - 1 = 0 \Rightarrow$ $n=4$, $m=1$, $B_1=3$ $\Rightarrow R_3 = 1 + \sqrt[3]{3/4} = 1{,}908$

$-1/R_3$ – верхня межа відємних коренів, тобто $-\frac{1}{R_3} = -0{,}5239$

Вкажемо межі тепер наших додатніх і відємних коренів вцілому:

$$2/3 \le x^+ \le 5$$

$$-2{,}587 \le x^- \le -0{,}5239$$

$$(R^+ = 5,\ R_+ = 2/3,\ R^- = -0{,}5239,\ R_- = -2{,}587)$$

Можна уточнити корені використовуючи спосіб Лагранжа

$f^+(x) = -x^4 + 3x^3 - 2x + 4 \Rightarrow x^4 - 3x^3 + 2x - 4 = 0$

$F^+(x) = x^4 - 3x^3 - 4$ де $f^+(x) = F^+(x) + \Phi^+(x)$

$\Phi^+(x) = 2x$

$f_+(x) = x^4\left(\frac{1}{x^4} - 3\frac{1}{x^3} + 2\cdot\frac{1}{x} - 4\right) = 1 - 3x + 2x^3 - 4x^4 = 0 \Rightarrow$

$\Rightarrow 4x^4 - 2x^3 + 3x - 1 = 0$

$F_+^+(x) = 4x^4 - 2x^3 - 1$ де $f_+(x) = F_+(x) + \Phi_+(x)$.

$\Phi_+^+(x) = 3x$.

Підберемо верхню та нижню межу для додатніх коренів:

$F^+(4) = 4^4 - 3\cdot 4^3 - 4 = 171 > 0$ $\qquad \underline{2{,}5 < x^+ < 4}$

$F_+(2{,}5) = 4\cdot 2{,}5^4 - 2\cdot 2{,}5^3 - 1 = 124 > 0$

Тепер знаючи межі в котрих лежать корені можемо розбити прямі на проміжки і застосувати теорему Штурма

Теорема Штурма говорить, що кількість дійсних коренів на $\mathbb{R}$ $[a;b]$ полінома $f_0(x)$ дорівнює різниці змін знаку у послідовності $(*)$ при $x=a$ та $x=b$, де $(*)$: $f_0=f(x)$, $f_1=f'(x)$, $f_{i+1}=-[f_{i-1} \bmod f_i]$, $i=\overline{1,m}$

Обчислимо:

$$f_0=-x^4+3x^3-2x+4.$$

$$f_1=-4x^3+9x^2-2.$$

$$f_2=-\frac{27}{16}x^2+\frac{3}{2}x-\frac{29}{8}.$$

$$f_3=-\frac{1088}{81}x+\frac{3328}{243}$$

$$f_4=\frac{8901}{2312}$$

Тоді знаки полінома Штурма

Многочлен	-4	-2	0	2	4	
f_0	−	−	+	+	−	
f_1	+	+	−	+	−	
f_2	−	−	−	−	−	
f_3	+	+	+	−	−	
f_4	+	+	+	+	+	
К зз	3	3	2	2	1	

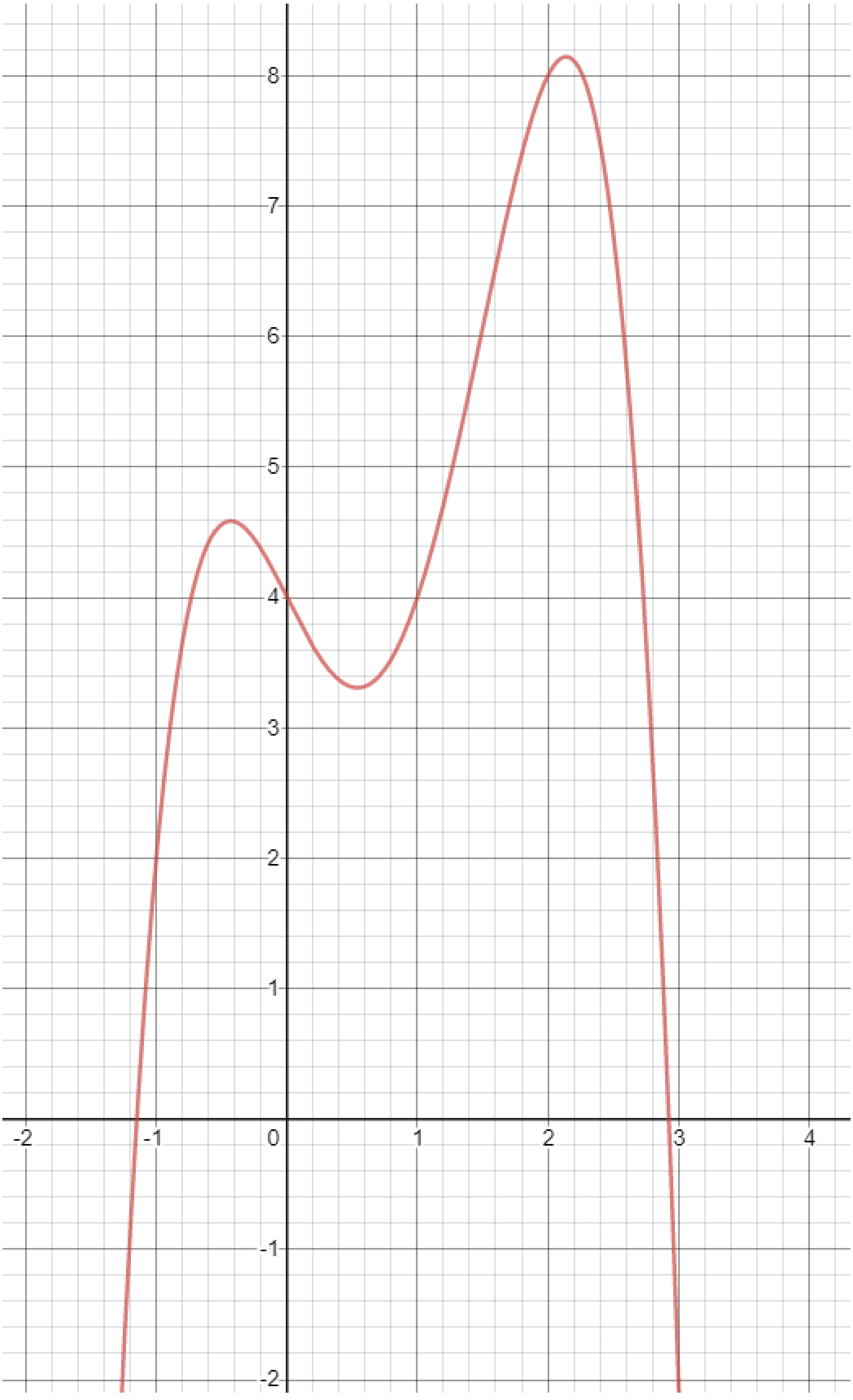
Отже є 2 проміжки де розміщено лише по одному кореню

(−2; 0) та (2; 4).

Оскільки межі додатніх і від'ємних коренів встановлено було раніше, то обмежимось остаточними значеннями

(2,5; 4) та (−2,59; −0,51)

Графік



Програмна частина

Код

```
#include <iostream>

#include <cmath>

#include <string>


#define line cout << endl << "________________________________________________________" << endl;


using namespace std;


double function_(double x) //OK

{

	double result = -pow(x, 4) + 3 * pow(x, 3) - 2 * x + 4;

	return result;

}


double derivative_(double x) //OK

{

	double result = -4 * (pow(x, 3)) + 9 * x * x - 2;


	return result;

}


double bisection_method(double a, double b, double epsilone) //OK

{

	cout << endl << " -method bisection :" << endl;


	int iteration = 1;


	double start = a;

	double end = b;


	while (end - start > epsilone)

	{

		if (function_((end + start) / 2) * function_(start) > 0)

		{

			start = (end + start) / 2;

		}

		else

		{

			end = (end + start) / 2;

		}
```

```
                cout << endl << "\t #"<<iteration++ << " iteration \t " << "  interval=" <<"["<< start << ";" << end <<
"]" << ";" << endl;

        }

        return (start + end) / 2;

}


double chord_method(double a, double b, double epsilone) //OK

{

        cout << endl << "-method chord: " << endl;


        int iteration = 1;


        double start = a;  double end = b;


        double middle_prev = (start * function_(end) - end * function_(start)) / (function_(end) - function_(start));


        double middle = middle_prev;


        do

        {

                middle_prev = middle;


                if (function_(middle_prev) * function_(start) > 0)

                {

                        start = middle_prev;

                }

                else

                {

                        end = middle_prev;

                }

                middle = (start * function_(end) - end * function_(start)) / (function_(end) - function_(start));


                cout << endl << "\t #" << iteration++ << " iteration \t " << "  interval=" <<"["<< start << ";" << end
<<"]"<< ";" << endl;

        }

        while (abs(middle - middle_prev) > epsilone);


        return middle;

}


double newton_method(double start, double epsilone) //  OK

{

        cout << endl << "-method newton : " << endl;
```

```
        int iteration = 1;

        double middle = start;

        double middle_prev = middle;

        do
        {
                middle_prev = middle;

                middle = middle - (function_(middle) / derivative_(middle));

                cout << endl << "\t #" << iteration++ << " iteration \t " << "  interval of step to the root=" <<"["<<
middle_prev << "; " << middle <<"]" << ";" << endl;

        }
        while (abs(middle - middle_prev) > epsilone);

        return middle;
}


int main()
{

        line

        double start_first = 2.5;  double end_first = 4;

        cout << endl << "Root of polynom (first):" << endl;

        cout << endl <<"-result of bisection: "<<bisection_method(start_first,end_first,0.00001)<<endl;
        cout << endl <<"-result of chord: "<<chord_method(start_first, end_first, 0.00001)<<endl;
        cout << endl <<"-result of newton: "<<newton_method(end_first, 0.00001)<<endl; //step from right to left (change
start point)

        line


        double start_second = -2.59;  double end_second = -0.51;

        cout << endl << "Root of polynom (second):" << endl;
```

```
	cout << endl << "-result of bisection: "<< bisection_method(start_second, end_second,0.00001)<<endl;

	cout << endl << "-result of chord: "<< chord_method(start_second, end_second,0.00001)<<endl;

	cout << endl << "-result of newton: "<< newton_method(start_second,0.00001)<<endl; //step from left to right
(change start point)

	line


	return 0;

}
```

Результати програми

```
Root of polynom (first):

 -method bisection :

         #1 iteration      interval=[2.5;3.25];

         #2 iteration      interval=[2.875;3.25];

         #3 iteration      interval=[2.875;3.0625];

         #4 iteration      interval=[2.875;2.96875];

         #5 iteration      interval=[2.92188;2.96875];

         #6 iteration      interval=[2.92188;2.94531];

         #7 iteration      interval=[2.92188;2.93359];

         #8 iteration      interval=[2.92188;2.92773];

         #9 iteration      interval=[2.9248;2.92773];

         #10 iteration     interval=[2.9248;2.92627];

         #11 iteration     interval=[2.92554;2.92627];

         #12 iteration     interval=[2.9259;2.92627];

         #13 iteration     interval=[2.9259;2.92609];

         #14 iteration     interval=[2.92599;2.92609];

         #15 iteration     interval=[2.92604;2.92609];

         #16 iteration     interval=[2.92606;2.92609];

         #17 iteration     interval=[2.92606;2.92607];

         #18 iteration     interval=[2.92607;2.92607];

-result of bisection: 2.92607

-method chord:

         #1 iteration      interval=[2.63659;4];

         #2 iteration      interval=[2.73668;4];

         #3 iteration      interval=[2.8056;4];

         #4 iteration      interval=[2.85092;4];

         #5 iteration      interval=[2.87979;4];

         #6 iteration      interval=[2.8978;4];

         #7 iteration      interval=[2.90889;4];

         #8 iteration      interval=[2.91567;4];

         #9 iteration      interval=[2.91978;4];

         #10 iteration     interval=[2.92227;4];

         #11 iteration     interval=[2.92378;4];
```

```
        #12 iteration     interval=[2.92469;4];
        #13 iteration     interval=[2.92524;4];
        #14 iteration     interval=[2.92557;4];
        #15 iteration     interval=[2.92577;4];
        #16 iteration     interval=[2.92589;4];
        #17 iteration     interval=[2.92596;4];
        #18 iteration     interval=[2.926;4];
        #19 iteration     interval=[2.92603;4];
        #20 iteration     interval=[2.92605;4];
-result of chord: 2.92606
-method newton :
        #1 iteration      interval of step to the root=[4; 3.40351];
        #2 iteration      interval of step to the root=[3.40351; 3.06598];
        #3 iteration      interval of step to the root=[3.06598; 2.94256];
        #4 iteration      interval of step to the root=[2.94256; 2.92634];
        #5 iteration      interval of step to the root=[2.92634; 2.92607];
        #6 iteration      interval of step to the root=[2.92607; 2.92607];
-result of newton: 2.92607
________________________________________________________
Root of polynom (second):
 -method bisection :
        #1 iteration      interval=[-1.55;-0.51];
        #2 iteration      interval=[-1.55;-1.03];
        #3 iteration      interval=[-1.29;-1.03];
        #4 iteration      interval=[-1.16;-1.03];
        #5 iteration      interval=[-1.16;-1.095];
        #6 iteration      interval=[-1.16;-1.1275];
        #7 iteration      interval=[-1.16;-1.14375];
        #8 iteration      interval=[-1.15187;-1.14375];
        #9 iteration      interval=[-1.15187;-1.14781];
        #10 iteration     interval=[-1.14984;-1.14781];
        #11 iteration     interval=[-1.14984;-1.14883];
        #12 iteration     interval=[-1.14934;-1.14883];
        #13 iteration     interval=[-1.14934;-1.14908];
        #14 iteration     interval=[-1.14934;-1.14921];
        #15 iteration     interval=[-1.14927;-1.14921];
```

```
        #16 iteration     interval=[-1.14927;-1.14924];

        #17 iteration     interval=[-1.14927;-1.14926];

        #18 iteration     interval=[-1.14927;-1.14926];

-result of bisection: -1.14927

-method chord:

        #1 iteration      interval=[-2.59;-0.612418];

        #2 iteration      interval=[-2.59;-0.706549];

        #3 iteration      interval=[-2.59;-0.790561];

        #4 iteration      interval=[-2.59;-0.863369];

        #5 iteration      interval=[-2.59;-0.92473];

        #6 iteration      interval=[-2.59;-0.975154];

        #7 iteration      interval=[-2.59;-1.01569];

        #8 iteration      interval=[-2.59;-1.04767];

        #9 iteration      interval=[-2.59;-1.07253];

        #10 iteration     interval=[-2.59;-1.09163];

        #11 iteration     interval=[-2.59;-1.10615];

        #12 iteration     interval=[-2.59;-1.11712];

        #13 iteration     interval=[-2.59;-1.12535];

        #14 iteration     interval=[-2.59;-1.13151];

        #15 iteration     interval=[-2.59;-1.1361];

        #16 iteration     interval=[-2.59;-1.13952];

        #17 iteration     interval=[-2.59;-1.14205];

        #18 iteration     interval=[-2.59;-1.14393];

        #19 iteration     interval=[-2.59;-1.14532];

        #20 iteration     interval=[-2.59;-1.14635];

        #21 iteration     interval=[-2.59;-1.14711];

        #22 iteration     interval=[-2.59;-1.14768];

        #23 iteration     interval=[-2.59;-1.14809];

        #24 iteration     interval=[-2.59;-1.1484];

        #25 iteration     interval=[-2.59;-1.14863];

        #26 iteration     interval=[-2.59;-1.1488];

        #27 iteration     interval=[-2.59;-1.14892];

        #28 iteration     interval=[-2.59;-1.14901];

        #29 iteration     interval=[-2.59;-1.14908];

        #30 iteration     interval=[-2.59;-1.14913];

        #31 iteration     interval=[-2.59;-1.14917];
```

```
#32 iteration     interval=[-2.59;-1.14919];

#33 iteration     interval=[-2.59;-1.14921];

#34 iteration     interval=[-2.59;-1.14923];

#35 iteration     interval=[-2.59;-1.14924];

-result of chord: -1.14925

-method newton :

#1 iteration      interval of step to the root=[-2.59; -1.90226];

#2 iteration      interval of step to the root=[-1.90226; -1.4558];

#3 iteration      interval of step to the root=[-1.4558; -1.2234];

#4 iteration      interval of step to the root=[-1.2234; -1.15495];

#5 iteration      interval of step to the root=[-1.15495; -1.14931];

#6 iteration      interval of step to the root=[-1.14931; -1.14927];

#7 iteration      interval of step to the root=[-1.14927; -1.14927];

-result of newton: -1.14927
```
