Лабораторна робота №1

Студентки групи ПМО-21

Кравець Олени

Варіант - 9.

Частина 1.

$x^4 - 3x^3 + 8x^2 - 5 = 0$

Крок 1. теорема 5.3. (Знаходж. верх. та нижн. межі додат. та відємн. коренів)

$x \le 1 + \sqrt[k]{\frac{B}{A_n}}$; $B = \max\limits_{A_i < 0} |A_i|$

$A_n = A_4 = 1$

$A_{n-k} = A_3 = -3$

$n - k = 3$

$k = 1$

$B = \max\{3; 5\} = 5$

$x = 1 + \sqrt[1]{\frac{5}{1}} = 1 + \sqrt{5}$

$a = 1 + \sqrt{5} \approx 3,23607$

Корені лежать у прямому $x \le 1 + \sqrt{5}$

Заміна: $z = \frac{1}{x}$

$z^4 - 3z^3 + 8z^2 - 5 = 0$

$A_n = A_4 = 1$

$A_{n-k} = A_3 = -3$

$n - k = 3$

$k = 1$

$B = \max\{3; 5\} = 5$

$b^* = 5 + 1 = 6$ – верхня межа додат. кор.

$\frac{1}{6}$ – нижня межа додат. кор.

$b = \frac{1}{6} \approx 0{,}16666$

Заміна: $u = -x$

$u^4 + 3u^3 - 8u^2 + 5 = 0$

$A_n = A_4 = 1$

$A_{n-k} = A_2 = -8$

$n - k = 2$; $k = 2$

$B = 8$

$c^* = 1 + \sqrt[2]{\frac{8}{1}} = 1 + 2\sqrt{2}$ – верх. межа

$c = -1 - 2\sqrt{2}$ – нижс межа

Заміна: $v = -\frac{1}{x}$

$v^4 + 3x^3 - 8x^2 + 5 = 0$

$A_n = A_4 = 1$

$A_{n-k} = A_2 = -8$

$n-k=2$

$k=2$

$B=8$

$d^+ = 1 + \sqrt[2]{\frac{8}{1}} = 1+2\sqrt{2}$ – верх. межа

$d = -\frac{1}{1+2\sqrt{2}}$ – нижн. межа

додатні корені є в інтерв.: $(\frac{1}{6}; 1+\sqrt{5})$ $0,16$ $3,23$

від'ємні корені є в інтерв.: $(-1-2\sqrt{2}; -\frac{1}{1+2\sqrt{2}})$ $-3,82$

Крок 2. $f(x) = x^4 - 3x^3 + 8x^2 - 5$ (Відкреслюємо графік)

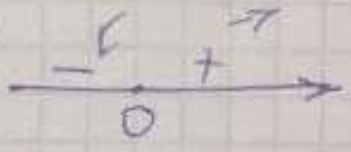
$f'(x) = 4x^3 - 9x^2 + 16x = 0$

$x(4x^2 - 9x + 16) = 0$

$x=0$ $\qquad 4x^2 - 9x + 16 = 0$

$D = 81 - 4 \cdot 4 \cdot 16 < 0$

$\varnothing$



$[a; B]$

$f(a) \cdot f(B) < 0$

Візьмемо $[-1; 0]$

$f(-1) +$

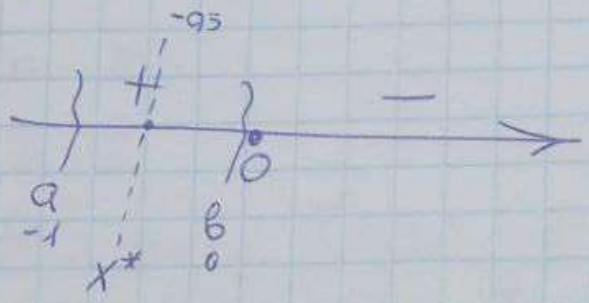
$f(0) -$

Візьмемо $[0; 1]$

$f(0) -$

$f(1) +$

$\varepsilon = 10^{-5} = 0{,}00001$ - точність



Крок 3- Використ. метод поділу відрізка навпіл знаходжу корені із заданою точністю $\varepsilon = 10^{-5}$

Результат виконання програми:

Microsoft Visual Studio Debug Console

```
a = -1
b = 0
e = 0.0001
Корені знайдено
a = -0.5          b = -0.5
a = -0.75         b = -0.75
a = -0.625        b = -0.625
a = -0.6875       b = -0.6875
a = -0.71875      b = -0.71875
a = -0.703125     b = -0.703125
a = -0.695313     b = -0.695313
a = -0.691406     b = -0.691406
a = -0.689453     b = -0.689453
a = -0.688477     b = -0.688477
a = -0.688965     b = -0.688965
a = -0.688721     b = -0.688721
a = -0.688843     b = -0.688843
a = -0.688782     b = -0.688782
a = -0.688751     b = -0.688751
Корінь x = -0.688736
Кількість кроків i = 16
```

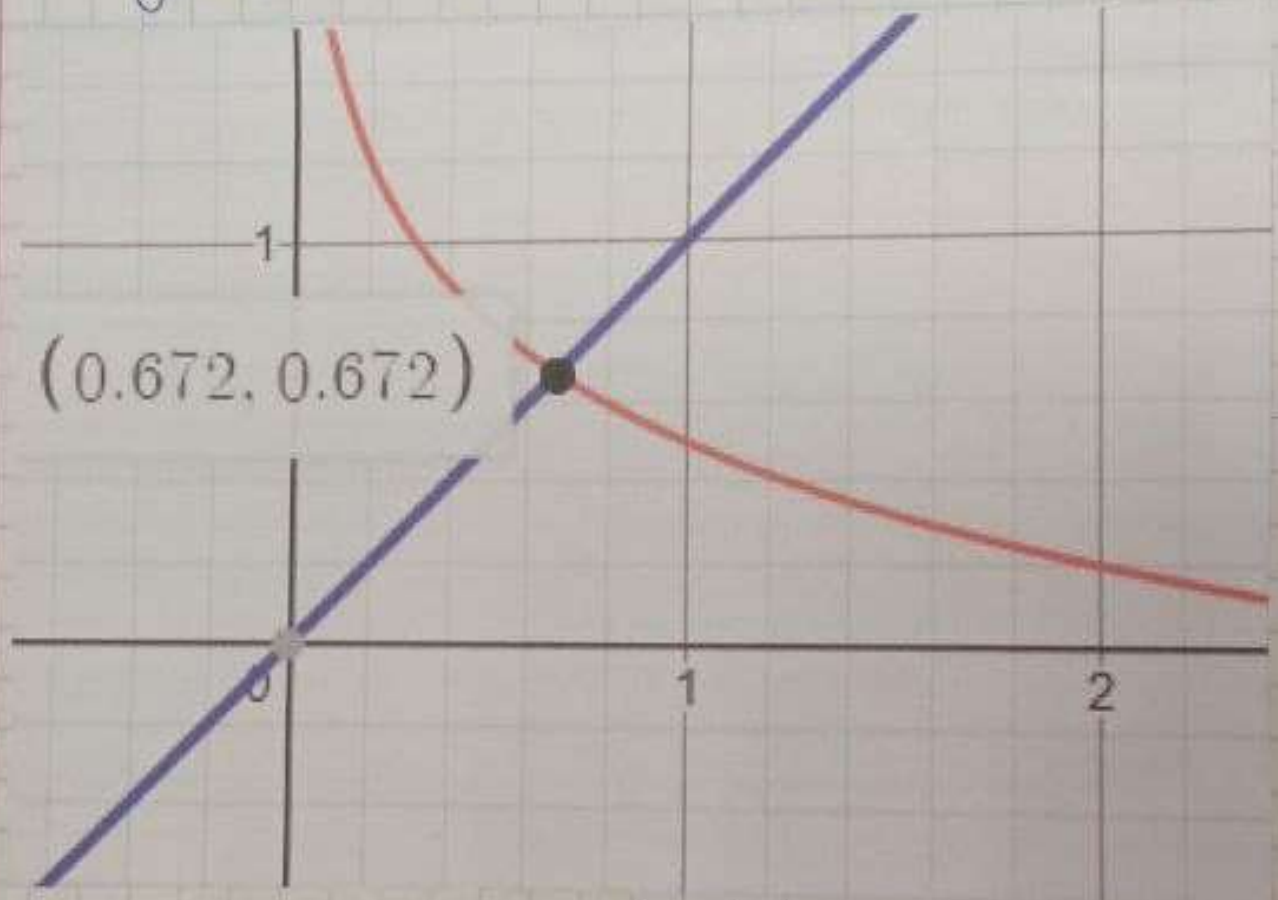
Код програми (C++):

```
#include <iostream>
#include <cmath>
#include <windows.h>
using namespace std;
// функція, яка містить рівняння y=x^4-3x^3+8x^2-5
double f(double x)
{
	double f;
	f = pow(x, 4) - 3 * pow(x, 3) + 8 * pow(x, 2) - 5;
	return f;
}
int main()
{
	// підключення укр.мови
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);

	double a = -1, b = 0, e = 0.0001, x; // відрізок [a; b] та задана точність
	int i = 0; // змінна для підрахунку кроків

	cout << "a = " << a << endl;
	cout << "b = " << b << endl;
	cout << "e = " << e << endl;

	// якщо виконується умова f(a)*f(b)<0, то шукаємо середину відрізка[a,b] x =(a+b)/2
	if (f(a) * f(b) < 0)
	{
		cout << "Корені знайдено" << endl;
		while (1)
		{
			x = (a + b) / 2.0;
			i = i + 1;
			if (fabs(f(x)) < e) break; // якщо |f(x)|<e, то виходимо з циклу
	// якщо f(a)*f(x)< 0, то шукаємо на якому кінці відрізка змінюється знак і замінюємо цей кінець на x
			if ((f(a) * f(x) < 0))
			{
				a = a;
				b = x;
			}
			else
			{
				a = x;
				b = b;
			}
			// виводимо значення a та b
			cout << "a = " << a << "\t";
			cout << "b = " << b << endl;
		}
		// виводимо значення x та i
		cout << "Корінь x = " << x << "\n";
		cout << "Кількість кроків i = " << i << "\n";
	}
	// якщо НЕ виконується умова f(a)*f(b)<0, то виводимо повідомлення
	else
	{
		cout << "Кореня не існує" << endl;
	}
	cout << endl;
	system("pause");
	return 0;
}
```

Частина 2.

Крок 1 Відокрем. дійсні корені рівняння графічним способом.

$lg(x) + x - 0,5 = 0$; $x = \varphi(x) \Rightarrow x = 0,5 - lg(x)$



$\varphi'(x) = (0,5 - lg(x))' = -\frac{1}{x \cdot \ln 10}$

$\varphi'(x) = \left|\frac{1}{x \cdot \ln 10}\right|$

$x^* = 1.$

Тому будемо шукати корені на інтервалі $(0,5; 1)$.

Використ. метод ітерації знаходжу корені заданого рівняння

Код програми (C++):

```cpp
#include <iostream>
#include <cmath>
#include <Windows.h>
using namespace std;

double f(double x)
{
    return x = 0.5 - log10(x);
}

int main()
{
    // підключення укр.мови
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    double a = 0.5, b = 1, e = 0.0001; // відрізок [a, b] та задана точність
    int i = 0, N = 1000; // i - змінна для підрахунку кроків ітерації, N - максимальна кількість ітерацій

    cout << "Інтервал (a, b): " << "("<< a << "," << b << ")" << endl;
    cout << "Задана точність e = " << e << "\n\n";

    while (i <= N) // якщо i<=N, то шукаємо корені
    {
        a = f(b); // за початкове наближення вибираємо 0,5
        if (fabs(a - b) < e) // якщо |a-b|<e, то виходимо з циклу
        {
            cout << "\nЗначення a = " << a << endl; // вивід значення a
            break;
        }
        cout << "Ітерація " << i << ": a = " << a << " \t"; // вивід кожної ітерації і  значення
        i++; // збільшення кількості кроків ітерації
        b = a;
        cout << "Результат: " << a << endl; // вивід результату
        // якщо кількість ітерацій більша ніж максмимальна к-сть ітер., то виводимо повідомлення
        if (i > N)
        {
            cout << "Коренів на даному інтервалі немає" << endl;;
            break;
        }
    }
    // вивід результату та на якому кроці він був знайдений
    cout << "\nКорінь x = " << a << " на ітерації " << i << endl;

    cout << endl;
    system("pause");
    return 0;
}
```

Результат виконання програми:

```
Інтервал (a, b): (0.5,1)
Задана точність e = 0.0001

Ітерація 0: a = 0.5          Результат: 0.5
Ітерація 1: a = 0.80103      Результат: 0.80103
Ітерація 2: a = 0.596351     Результат: 0.596351
Ітерація 3: a = 0.724498     Результат: 0.724498
Ітерація 4: a = 0.639963     Результат: 0.639963
Ітерація 5: a = 0.693845     Результат: 0.693845
Ітерація 6: a = 0.658737     Результат: 0.658737
Ітерація 7: a = 0.681288     Результат: 0.681288
Ітерація 8: a = 0.666669     Результат: 0.666669
Ітерація 9: a = 0.676089     Результат: 0.676089
Ітерація 10: a = 0.669996    Результат: 0.669996
Ітерація 11: a = 0.673928    Результат: 0.673928
Ітерація 12: a = 0.671387    Результат: 0.671387
Ітерація 13: a = 0.673027    Результат: 0.673027
Ітерація 14: a = 0.671967    Результат: 0.671967
Ітерація 15: a = 0.672652    Результат: 0.672652
Ітерація 16: a = 0.67221     Результат: 0.67221
Ітерація 17: a = 0.672495    Результат: 0.672495
Ітерація 18: a = 0.672311    Результат: 0.672311
Ітерація 19: a = 0.67243     Результат: 0.67243

Значення a = 0.672353

Корінь x = 0.672353 на ітерації 20
```

Використавши метод Ньютона (метод дотичних) знаходжу корені заданого рівняння.

Код програми (c++):

```cpp
#include <iostream>
#include <cmath>
#include <Windows.h>
using namespace std;
// функція, яка містить рівняння log10(x) +x-0.5
double f(double x)
{
	return log10(x) + x - 0.5;
}

// похідна від log10(x)+x-0.5
double p(double x)
{
	return (1 / x * (log10(10) / log10(exp(x))) + 1); // (1/x*ln(10))+1
}

// друга похідна від log10(x)+x-0.5
double p2(double x)
{
	return -1 / x * x * (log10(10) / log10(exp(x))); // -1 / x^2 * ln(10)
}

int main()
{
	// підключення укр.мови
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);

	double a = 0.5, b = 1, e = 0.0001; // відрізок [a, b] та задана точність

	cout << "Інтервал (a, b): " << "(" << a << "," << b << ")" << endl;
	cout << "Задана точність e = " << e << "\n\n";

	double x, x_0, x_1, x_2, k;
	int i = 0; // кількість ітерацій
	if (f(a) * f(b) < 0)
	{
		// заміна a на b
		if (f(a) * p2(a) > 0)
		{
			x_0 = a;
		}
		else
		{
			x_0 = b;
		}
		while (1)
		{

			x_1 = x_0 - (f(x_0) / p(x_0));
			x_2 = x_1 - (f(x_1) / p(x_1));

			if (abs(x_2 - x_1) < e)
			{
				break;
			}
			x_0 = x_1;
			x_1 = x_0- (f(x_0) / p(x_0));
			x_1 = x_2;
			x_2 = x_1 - (f(x_1) / p(x_1));
			i++;
			cout << "Ітерація " << i << ": " << "\t" << "x_0 = " << x_0 << endl;
		}
		cout << "\nКількість ітерацій i = " << i << "\n"; // виводимо кількість ітерацій i
		cout << "\nКорінь x = " << x_2 << " на ітерації " << i << endl;
		return x_2;
	}
	else
	{
		cout << "Коренів не існує" << endl;
	}
	cout << endl;
	system("pause");
	return 0;
}
```

Результат виконання програми:

```
Інтервал (a, b): (0.5,1)
Задана точність e = 0.0001

Ітерація 1:     x_0 = 0.529483
Ітерація 2:     x_0 = 0.556256
Ітерація 3:     x_0 = 0.579767
Ітерація 4:     x_0 = 0.599763
Ітерація 5:     x_0 = 0.616282
Ітерація 6:     x_0 = 0.629583
Ітерація 7:     x_0 = 0.640064
Ітерація 8:     x_0 = 0.648177
Ітерація 9:     x_0 = 0.654369
Ітерація 10:    x_0 = 0.659043
Ітерація 11:    x_0 = 0.662541
Ітерація 12:    x_0 = 0.665142
Ітерація 13:    x_0 = 0.667067
Ітерація 14:    x_0 = 0.668487
Ітерація 15:    x_0 = 0.66953
Ітерація 16:    x_0 = 0.670296
Ітерація 17:    x_0 = 0.670857
Ітерація 18:    x_0 = 0.671268
Ітерація 19:    x_0 = 0.671569
Ітерація 20:    x_0 = 0.671788
Ітерація 21:    x_0 = 0.671949

Кількість ітерацій i = 21

Корінь x = 0.672152 на ітерації 21
```

Використавши метод хорд знайдемо корені заданого рівняння

Код програми (c++):

```cpp
#include <iostream>
#include <cmath>
#include <Windows.h>
using namespace std;

// функція, яка містить рівняння log10(x) + x - 0.5
double f(double x)
{
	return log10(x) + x - 0.5;
}

// (0.5 - log10(x))' = 1 / x * ln(10)
// (0.5 - log10(x))'' = (1 / x * ln(10))' = -1 / x^2 * ln(10)

// друга похідна від 0.5 - log10(x)
double p2(double x)
{
	return -1 / x * x * (log10(10) / log10(exp(x))); // -1 / x^2 * ln(10)
}

int main()
{
	// підключення укр.мови
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);

	double a = 0.5, b = 1, e = 0.0001; // відрізок [a, b] та задана точність

	cout << "Інтервал (a, b): " << "(" << a << "," << b << ")" << endl;
	cout << "Задана точність e = " << e << "\n\n";

	int i = 0; // кількість ітерацій
	double x_0;
	if (f(a) * f(b) < 0)
	{
		while (fabs(b - a) >= e)
		{
			x_0 = (b * f(a) - f(b) * a) / (f(a) - f(b));
			if (f(x_0) * f(b) < 0)
			{
				a = x_0;
			}
			else if (f(a) * f(x_0) < 0)
			{
				b = x_0;
			}
			else
				return x_0;
			i++;
			cout << "Ітерація " << i << ": " << "\t" << "x_0 = " << x_0 << endl;
		}
		cout << "\nКількість ітерацій i = " << i << "\n"; // виводимо кількість ітерацій i
		cout << "\nКорінь x = " << x_0 << " на ітерації " << i << endl;
		return x_0;

	}
	else
	{
		cout << "\nКоренів не існує" << endl;
	}

	cout << endl;
	system("pause");
	return 0;
}
```

Результат виконання програми:

```
Інтервал (a, b): (0.5,1)
Задана точність e = 0.0001

Ітерація 1:     x_0 = 0.687902
Ітерація 2:     x_0 = 0.673266
Ітерація 3:     x_0 = 0.672434
Ітерація 4:     x_0 = 0.672386
Ітерація 5:     x_0 = 0.672383
Ітерація 6:     x_0 = 0.672383
Ітерація 7:     x_0 = 0.672383
Ітерація 8:     x_0 = 0.672383
Ітерація 9:     x_0 = 0.672383
Ітерація 10:    x_0 = 0.672383
Ітерація 11:    x_0 = 0.672383
Ітерація 12:    x_0 = 0.672383
Ітерація 13:    x_0 = 0.672383
Ітерація 14:    x_0 = 0.672383

Кількість ітерацій i = 14

Корінь x = 0.672383 на ітерації 14
```