МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАВЧАЛЬНО-НАУКОВИЙ КОМПЛЕКС
«ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ»
НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Лабораторна робота №1

з курсу «Чисельні методи»

тема: «МЕТОДИ РОЗВ'ЯЗАННЯ НЕЛІНІЙНИХ АЛГЕБРАЇЧНИХ РІВНЯНЬ»

Виконав: студент 2 курсу
групи КА-23
Деундяк О.В.
Прийняла: Кузнєцова Н. В.

Київ – 2014р.

Варіант 9

Умова: Знайти всі дійсні корені рівняння: $x^4 - 2x^3 - 9x^2 - 3x - 3 = 0$ попередньо відокремивши їх, а потім із застосуванням чисельних методів уточнити розв'язки.

Допрограмовий етап (відокремлення коренів).

Для визначення кількості коренів (з умовою що кратних коренів немає) та їх відокремлення використаємо метод Штурма.

Побудуємо ряд Штурма:

$$f_0(x) = x^4 - 2x^3 - 9x^2 - 3x - 3$$

$$f_1(x) = 4x^3 - 6x^2 - 18x - 3$$

$$f_2(x) = 42x^2 + 36x + 27$$

$$f_3(x) = 612x - 150$$

$$f_4(x) = -1$$

Знайдемо кількість дійсних коренів. Для цього дослідимо скільки змін знаків втрачає система Штурма при переході від $-\infty$ до $+\infty$.

	$f_0(x)$	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	
$-\infty$	+	-	+	-	-	3
$+\infty$	+	+	+	+	-	1

Отже, за теоремою Штурма рівняння має 3-1=2 дійсних корені.

За теоремою про обмеженість коренів

$$x \le 1 + \sqrt[4-3]{\frac{9}{1}} = 10$$

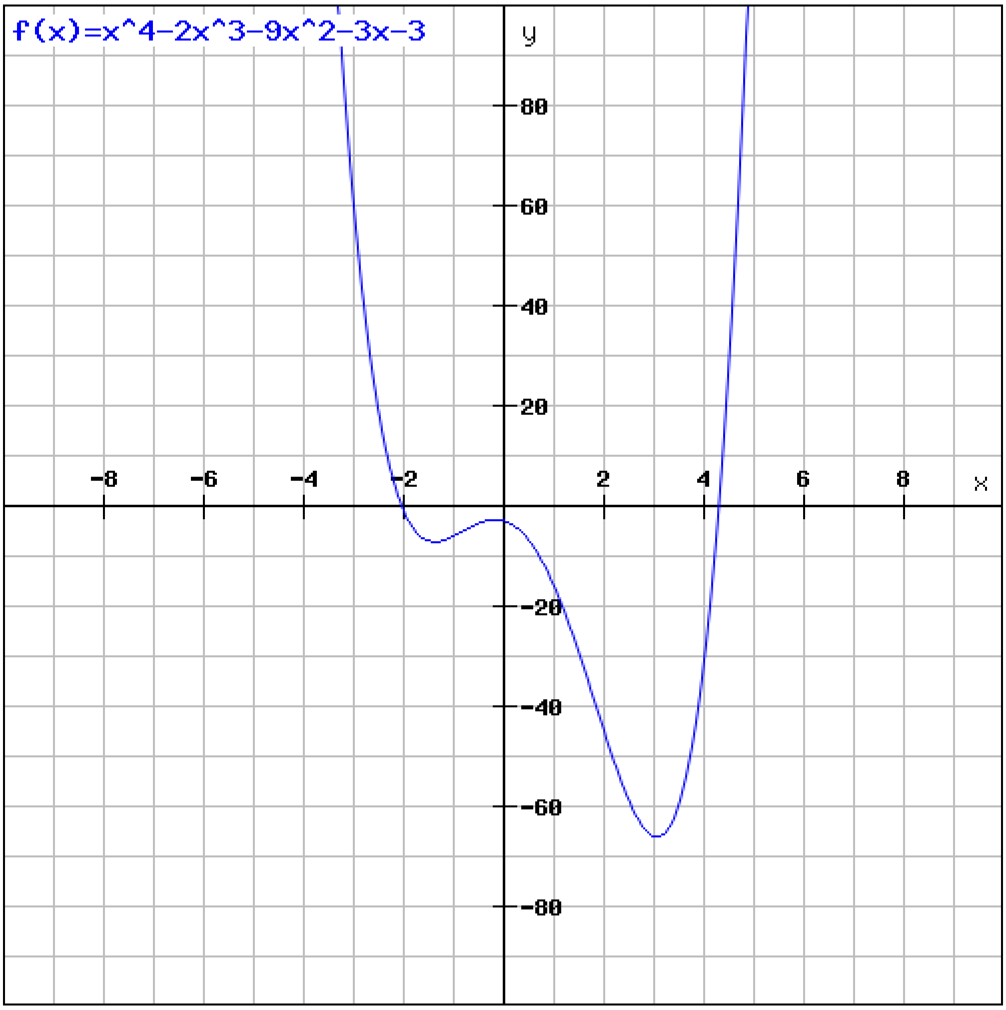
$$x \ge -\left(1 + \sqrt[4-3]{\frac{9}{1}}\right) = -10$$

Локалізуємо корені:

	$f_0(x)$	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	
-10	+	-	+	-	-	3
0	-	-	+	-	-	2
10	+	+	+	+	-	1

На відрізку $[-10\,;0]$ та $[0\,;10]$ поліном має по 1 кореню.

Скористаємось графічним методом для знаходження проміжків де зберігається знак першої та другої похідної.



Це відрізки $[-3\,;-1.8]$ та $[3.5{,}5]$.

Текст програми:

Polymomial.h

```
#pragma once

#include <fstream>

const long MAXIT = 5000;

class Polynomial
{
	double m_a5, m_a4, m_a3, m_a2, m_a1, m_a0;
	double Value(double x) const;
	Polynomial Derivative() const;
	Polynomial();
	mutable std::ofstream debug;

	public:
		Polynomial(double a5, double a4, double a3, double a2, double a1,
			   double a0, const std::string &debugfile = "debug.txt");
		~Polynomial();
		double GetRootBin(double left, double right, double acc) const;
		double GetRootChord(double left, double right, double acc) const;
		double GetRootNewt(double left, double right, double acc) const;
};
```

Polymomial.cpp

```
#include "Polynomial.h"
#include "cmath"
#include <string>

using std::endl;

template <typename T>
inline bool sgn(T val)
{
	return (val >= T(0));
}

Polynomial::Polynomial(double a5, double a4, double a3, double a2, double a1, double a0,
const std::string &debugfile) :
	m_a5(a5), m_a4(a4),
	m_a3(a3), m_a2(a2),
	m_a1(a1), m_a0(a0)
{
	debug.open(debugfile, std::ios_base::out);
}


Polynomial::~Polynomial()
{
	debug.close();
}

inline double Polynomial::Value(double x) const
{
	return m_a5*pow(x, 5) + m_a4*pow(x, 4) + m_a3*pow(x, 3) + m_a2*pow(x, 2)
		+ m_a1*x + m_a0;
}
```

```cpp
double Polynomial::GetRootBin(double left, double right, double acc) const
{
	debug << "Bisection method" << endl;

	long it = 0;
	double mid, midvalue;

	if (sgn(Value(left)) == sgn(Value(right)))
		throw std::runtime_error("No root or there are more then 1");

	do
	{
		++it;
		mid = (left + right) / 2;
		midvalue = Value(mid);
		if (midvalue * Value(left) > 0)
			left = mid;
		else
			right = mid;
		debug << "Iteration " << it << " x=" << mid << " f(x)=" << midvalue
			<< endl;
	} while (right - left > acc || abs(midvalue) > acc);

	debug << "Result: x = " << mid << endl << endl;
	return mid;
}

double Polynomial::GetRootChord(double left, double right, double acc) const
{
	debug << "Chord method" << endl;
	long it = 1;
	double s = right, c0 = left, c, svalue, cvalue;

	if (sgn(Value(left)) == sgn(Value(right)))
		throw std::runtime_error("No root or there are more then 1");

	c = (left * Value(right) - right * Value(left)) / (Value(right) - Value(left));
	cvalue = Value(c);
	if (sgn(Value(left)) != sgn(Value(c)))
	{
		c0 = right;
		s = left;
	}
	svalue = Value(s);

	while (abs(c - c0) > acc || abs(cvalue) > acc)
	{
		++it;
		c0 = c;
		c = (c * svalue - s * cvalue) / (svalue - cvalue);
		cvalue = Value(c);
		debug << "Iteration " << it << " x=" << c << " f(x)=" << cvalue << endl;
		if (it > MAXIT)
			throw std::exception(“Too many iterations”);
	}

	debug << "Result: x = " << c << endl << endl;
	return c;
}

double Polynomial::GetRootNewt(double left, double right, double acc) const
{
	debug << "Newton's method" << endl;
	long it = 1;
	Polynomial derivative = Derivative();
```

```
	double c0, c, cvalue;

	if (sgn(Value(left)) == sgn(Value(right)))
		throw std::runtime_error("No root or there are more then 1");

	c = left - Value(left) / derivative.Value(left);
	c0 = left;
	if (c > right)
	{
		c = right - Value(right) / derivative.Value(right);
		c0 = right;
	}

	cvalue = Value(c);

	while (abs(c - c0) > acc || abs(cvalue) > acc)
	{
		++it;
		c0 = c;
		c = c0 - Value(c) / derivative.Value(c);
		cvalue = Value(c);
		debug << "Iteration " << it << " x=" << c << " f(x)=" << cvalue << endl;
	}

	debug << "Result: x = " << c << endl << endl;
	return c;
}

Polynomial Polynomial::Derivative() const
{
	return Polynomial(0, 5 * m_a5, 4 * m_a4, 3 * m_a3, 2 * m_a2, m_a1, "debug1.txt");
}
```

main.cpp

```
#include "Polynomial.h"
#include <iostream>

const double eps = 1E-05;

using std::endl;
using std::cout;

int main()
{
	Polynomial pol(0, 1, -2, -9, -3, -3);
	//Polynomial pol(16, -3, 1, 2, -4, 2);

	double a = -10, b = 0;

	cout.setf(std::ios::scientific);
	try
	{
		cout << pol.GetRootBin(a, b, eps) << endl;
	}
	catch (std::exception e)
	{
		cout << e.what() << endl;
	}

	try
	{
		cout << pol.GetRootChord(a, b, eps) << endl;
```

```
	}
	catch (std::exception e)
	{
		cout << e.what() << endl;
	}

	try
	{
		cout << pol.GetRootNewt(a, b, eps) << endl;
	}
	catch (std::exception e)
	{
		cout << e.what() << endl;
	}

	cout.unsetf(std::ios::scientific);


	std::cin.get();
	std::cin.get();

	return 0;
}
```

Результати роботи програми

На [-3.5,-1.8]:

Bisection method

Iteration 1 [-2.65, -1.8] f1=28.28226 f2=-4.5984

Iteration 2 [-2.225, -1.8] f1=5.658344 f2=-4.5984

Iteration 3 [-2.225, -2.0125] f1=5.658344 f2=-0.7082617

Iteration 4 [-2.11875, -2.0125] f1=2.128937 f2=-0.7082617

Iteration 5 [-2.065625, -2.0125] f1=0.6284991 f2=-0.7082617

Iteration 6 [-2.065625, -2.039063] f1=0.6284991 f2=-0.05976539

Iteration 7 [-2.052344, -2.039063] f1=0.2793243 f2=-0.05976539

Iteration 8 [-2.045703, -2.039063] f1=0.1085278 f2=-0.05976539

Iteration 9 [-2.042383, -2.039063] f1=0.02406941 f2=-0.05976539

Iteration 10 [-2.042383, -2.040723] f1=0.02406941 f2=-0.0179258

Iteration 11 [-2.041553, -2.040723] f1=0.003052337 f2=-0.0179258

Iteration 12 [-2.041553, -2.041138] f1=0.003052337 f2=-0.007441597

Iteration 13 [-2.041553, -2.041345] f1=0.003052337 f2=-0.002195847

Iteration 14 [-2.041449, -2.041345] f1=0.0004279406 f2=-0.002195847

Iteration 15 [-2.041449, -2.041397] f1=0.0004279406 f2=-0.0008840292

Iteration 16 [-2.041449, -2.041423] f1=0.0004279406 f2=-0.0002280633

Iteration 17 [-2.041436, -2.041423] f1=9.993392e-005 f2=-0.0002280633

Iteration 18 [-2.041436, -2.04143] f1=9.993392e-005 f2=-6.406587e-005

Iteration 19 [-2.041433, -2.04143] f1=1.793372e-005 f2=-6.406587e-005

Iteration 20 [-2.041433, -2.041431] f1=1.793372e-005 f2=-2.306615e-005

Iteration 21 [-2.041433, -2.041432] f1=1.793372e-005 f2=-2.566231e-006

Result: x = -2.041432 in 21 iterations

Chord method
Iteration 1 [-3.5, -1.856786] f1=133.0625 f2=-3.76909
Iteration 2 [-3.5, -1.90205] f1=133.0625 f2=-3.00312
Iteration 3 [-3.5, -1.937318] f1=133.0625 f2=-2.338047
Iteration 4 [-3.5, -1.964302] f1=133.0625 f2=-1.787101
Iteration 5 [-3.5, -1.984654] f1=133.0625 f2=-1.346639
Iteration 6 [-3.5, -1.999836] f1=133.0625 f2=-1.00377
Iteration 7 [-3.5, -2.011068] f1=133.0625 f2=-0.7421161
Iteration 8 [-3.5, -2.019326] f1=133.0625 f2=-0.5453462
Iteration 9 [-3.5, -2.02537] f1=133.0625 f2=-0.3989574
Iteration 10 [-3.5, -2.029778] f1=133.0625 f2=-0.2909058
Iteration 11 [-3.5, -2.032985] f1=133.0625 f2=-0.2116091
Iteration 12 [-3.5, -2.035314] f1=133.0625 f2=-0.1536582
Iteration 13 [-3.5, -2.037004] f1=133.0625 f2=-0.1114357
Iteration 14 [-3.5, -2.038228] f1=133.0625 f2=-0.08074046
Iteration 15 [-3.5, -2.039114] f1=133.0625 f2=-0.05846114
Iteration 16 [-3.5, -2.039756] f1=133.0625 f2=-0.04230898
Iteration 17 [-3.5, -2.04022] f1=133.0625 f2=-0.03060872
Iteration 18 [-3.5, -2.040556] f1=133.0625 f2=-0.02213846
Iteration 19 [-3.5, -2.040799] f1=133.0625 f2=-0.01600921
Iteration 20 [-3.5, -2.040974] f1=133.0625 f2=-0.01157536
Iteration 21 [-3.5, -2.041101] f1=133.0625 f2=-0.008368683
Iteration 22 [-3.5, -2.041193] f1=133.0625 f2=-0.006049921
Iteration 23 [-3.5, -2.041259] f1=133.0625 f2=-0.004373413
Iteration 24 [-3.5, -2.041307] f1=133.0625 f2=-0.003161371
Iteration 25 [-3.5, -2.041342] f1=133.0625 f2=-0.002285172
Iteration 26 [-3.5, -2.041367] f1=133.0625 f2=-0.001651788
Iteration 27 [-3.5, -2.041385] f1=133.0625 f2=-0.001193943
Iteration 28 [-3.5, -2.041398] f1=133.0625 f2=-0.0008629957
Iteration 29 [-3.5, -2.041407] f1=133.0625 f2=-0.0006237787
Iteration 30 [-3.5, -2.041414] f1=133.0625 f2=-0.0004508688
Iteration 31 [-3.5, -2.041419] f1=133.0625 f2=-0.0003258879
Iteration 32 [-3.5, -2.041423] f1=133.0625 f2=-0.0002355511
Iteration 33 [-3.5, -2.041425] f1=133.0625 f2=-0.0001702555
Iteration 34 [-3.5, -2.041427] f1=133.0625 f2=-0.0001230599
Iteration 35 [-3.5, -2.041429] f1=133.0625 f2=-8.894701e-005
Iteration 36 [-3.5, -2.04143] f1=133.0625 f2=-6.429038e-005
Iteration 37 [-3.5, -2.04143] f1=133.0625 f2=-4.646868e-005
Iteration 38 [-3.5, -2.041431] f1=133.0625 f2=-3.358726e-005
Iteration 39 [-3.5, -2.041431] f1=133.0625 f2=-2.427665e-005
Iteration 40 [-3.5, -2.041431] f1=133.0625 f2=-1.754699e-005
Iteration 41 [-3.5, -2.041432] f1=133.0625 f2=-1.268284e-005
Iteration 42 [-3.5, -2.041432] f1=133.0625 f2=-9.16707e-006

Result: x = -2.041432 in 42 iterations

Newton's method
Iteration 1 x=-2.143986 f(x)=2.901711
Iteration 2 x=-2.051601 f(x)=0.2601038
Iteration 3 x=-2.041546 f(x)=0.00287781
Iteration 4 x=-2.041432 f(x)=3.657131e-007
Iteration 5 x=-2.041432 f(x)=1.065814e-014
Result: x = -2.041432 in 5 iterations

Ha [3.5,5]:

Bisection method
Iteration 1 [4.25, 5] f1=-5.589844 f2=132
Iteration 2 [4.25, 4.625] f1=-5.589844 f2=50.30493
Iteration 3 [4.25, 4.4375] f1=-5.589844 f2=19.45509
Iteration 4 [4.25, 4.34375] f1=-5.589844 f2=6.245713
Iteration 5 [4.25, 4.296875] f1=-5.589844 f2=0.1609431
Iteration 6 [4.273438, 4.296875] f1=-2.755613 f2=0.1609431
Iteration 7 [4.285156, 4.296875] f1=-1.307698 f2=0.1609431
Iteration 8 [4.291016, 4.296875] f1=-0.5759776 f2=0.1609431
Iteration 9 [4.293945, 4.296875] f1=-0.2081684 f2=0.1609431
Iteration 10 [4.29541, 4.296875] f1=-0.02377558 f2=0.1609431
Iteration 11 [4.29541, 4.296143] f1=-0.02377558 f2=0.06854301
Iteration 12 [4.29541, 4.295776] f1=-0.02377558 f2=0.02237353
Iteration 13 [4.295593, 4.295776] f1=-0.000703571 f2=0.02237353
Iteration 14 [4.295593, 4.295685] f1=-0.000703571 f2=0.01083434
Iteration 15 [4.295593, 4.295639] f1=-0.000703571 f2=0.005065226
Iteration 16 [4.295593, 4.295616] f1=-0.000703571 f2=0.002180788
Iteration 17 [4.295593, 4.295605] f1=-0.000703571 f2=0.0007385985
Iteration 18 [4.295593, 4.295599] f1=-0.000703571 f2=1.751125e-005
Iteration 19 [4.295596, 4.295599] f1=-0.0003430305 f2=1.751125e-005
Iteration 20 [4.295598, 4.295599] f1=-0.0001627598 f2=1.751125e-005
Iteration 21 [4.295598, 4.295599] f1=-7.262431e-005 f2=1.751125e-005
Iteration 22 [4.295599, 4.295599] f1=-2.755654e-005 f2=1.751125e-005
Iteration 23 [4.295599, 4.295599] f1=-5.022651e-006 f2=1.751125e-005
Result: x = 4.295599 in 23 iterations

Chord method
Iteration 1 [3.96572, 5] f1=-33.84034 f2=132
Iteration 2 [4.176769, 5] f1=-13.92779 f2=132
Iteration 3 [4.25534, 5] f1=-4.951236 f2=132
Iteration 4 [4.282262, 5] f1=-1.667195 f2=132
Iteration 5 [4.291214, 5] f1=-0.5510693 f2=132
Iteration 6 [4.294161, 5] f1=-0.18103 f2=132

Iteration 7 [4.295128, 5] f1=-0.05934915 f2=132
Iteration 8 [4.295445, 5] f1=-0.01944418 f2=132
Iteration 9 [4.295548, 5] f1=-0.006368984 f2=132
Iteration 10 [4.295582, 5] f1=-0.002086026 f2=132
Iteration 11 [4.295593, 5] f1=-0.0006832176 f2=132
Iteration 12 [4.295597, 5] f1=-0.0002237665 f2=132
Iteration 13 [4.295598, 5] f1=-7.328752e-005 f2=132
Iteration 14 [4.295599, 5] f1=-2.400295e-005 f2=132
Iteration 15 [4.295599, 5] f1=-7.861385e-006 f2=132
Result: x = 4.295599 in 15 iterations

Newton's method
Iteration 1 x=4.486381 f(x)=26.91295
Iteration 2 x=4.314611 f(x)=2.423476
Iteration 3 x=4.295813 f(x)=0.02703955
Iteration 4 x=4.295599 f(x)=3.494694e-006
Iteration 5 x=4.295599 f(x)=1.776357e-014
Result: x = 4.295599 in 5 iterations

Висновки:

Порівняємо всі методи за кількістю ітерацій:

Назва методу	1 корінь	2 корінь
Метод бісекції	21	23
Метод хорд	42	15
Метод Ньютона	5	5

Як бачимо, для заданого полінома найкраще спрацював метод Ньютона.