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

Варіант 4

Лабораторна робота №4

Методи обчислень
«Обчислення власних значень»

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

Викладач:
Стьопочкіна Ірина Валеріївна

Київ – 2021

Теорія

Метод А.Н. Кирилова

Власним вектором x_i матриці A, якому відповідає власне значення λ_i, називається ненульовий розв'язок системи рівнянь $(A-\lambda_i I)x_i=0$. Рівняння $\det(A-\lambda I)=0$ називається характеристичним. Коренями цього рівняння є власні числа.

Метод базується на теоремі Гамільтона-Келі про те, що кожна матриця є коренем свого характеристичного полінома, тобто:

$$(-1)^n A^n + p_1 A^{n-1} + \ldots + p_n I = 0.$$

Тому для довільного вектора $y_0 \neq 0$ виконується рівність:

$$(-1)^n A^n y_0 + p_1 A^{n-1} y_0 + \ldots + p_n y_0 = 0.$$

Це співвідношення представляє собою систему лінійних рівнянь відносно невідомих $p_1, p_2, \ldots, p_n$- коефіцієнтів характеристичного полінома.

Позначимо $A^k y_0 = y_k = (y_{1k}, y_{2k}, \ldots, y_{nk})'$.

Одержуємо систему відносно невідомих $p_1, \ldots, p_n$., використовуючи:

$$\left.\begin{array}{l} y_{1n-1}p_1 + y_{1n-2}p_2 + \ldots + p_n y_{10} = (-1)^{n+1} y_{1n} \\ y_{2n-1}p_1 + y_{2n-2}p_2 + \ldots + p_n y_{20} = (-1)^{n+1} y_{2n} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ y_{nn-1}p_1 + y_{nn-2}p_2 + \ldots + p_n y_{n0} = (-1)^{n+1} y_{nn} \end{array}\right\}$$

Таким чином, задають довільний вектор y_0 і будують вектори $y_k = Ay_{k-1}$. Наприклад, для тривимірного випадку можна задати $y_0=(1,0,0)^T$. Початковий вектор потрібно задавати так, щоб система була невиродженою. Метод не працює, якщо матриця A має кратні власні значення. Результатом розв'язання системи є значення коефіцієнтів, за якими будують характеристичний поліном та розв'язують відповідне рівняння, знаходячи власні числа.

Практика

Вхідні данні

Матриця та метод, що вказаний в умові:

6,3	1,07	0,99	1,20	Крилова
1,07	4,12	1,30	0,16	
0,99	1,30	5,48	2,10	
1,20	0,16	2,10	6,06	

Скористаємося рівністю Гамільтона-Келі:

$A^4 + p_1A^3 + p_2A^2 + p_3A + p_4I = 0$, де p_i- коєфіцієнти характеристичного полінома.

Домножимо рівність на вектор $y_0 = (0, 0, 0, 1)^T$.

Знаходимо вектори: $y_k = Ay_{k-1}$.

Отримані вектори ми заносимо з кінця матриці нижченаведеної G, останній обчислений вектор, буде вектором вже B, але з відповідним протилежним знаком.

Отримаємо **СЛАР**: $Gx=B$

Матриця G:

$$\begin{matrix} 190.148 & 17.0822 & 1.2 & 0 \\ 81.6612 & 5.6428 & 0.16 & 0 \\ 254.158 & 25.63 & 2.1 & 0 \\ 333.376 & 42.5992 & 6.06 & 1 \end{matrix}$$

Вектор стовпчик B:

$$\begin{matrix} -1936.98 \\ -923.648 \\ -2387.28 \\ -2795.23 \end{matrix}$$

Розв’язуємо систему:

$$190.148*x_0 + 17.0822*x_1 + 1.2*x_2 + 0*x_3 = -1936.98$$

$$81.6612*x_0 + 5.6428*x_1 + 0.16*x_2 + 0*x_3 = -923.648$$

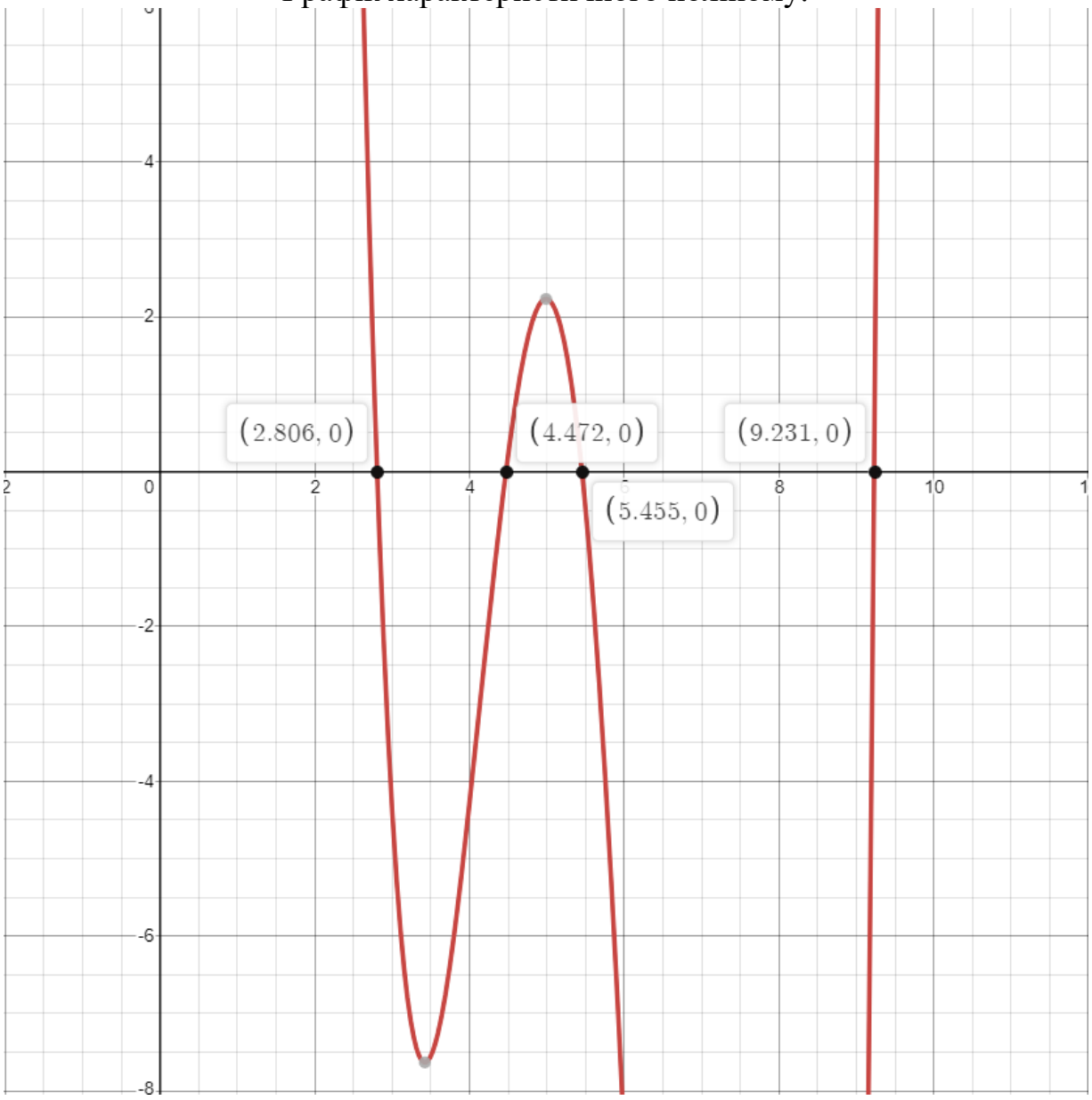
$$254.158*x_0 + 25.63*x_1 + 2.1*x_2 + 0*x_3 = -2387.28$$

$$333.376*x_0 + 42.5992*x_1 + 6.06*x_2 + 1*x_3 = -2795.23$$

Розв’язок X:

$$-21.96$$
$$169.721$$
$$-550.44$$
$$631.388$$

Графік характеристичного поліному:



За теоремою про верхню межу:

$$1.85<x^{+}<551,24$$

За допомогою метода бісекції знайдемо корені характеристичного полінома(власні числа):

$$\lambda_1=2.80615;$$

$$\lambda_2=4.47179;$$

$$\lambda_3=5.45465;$$

$$\lambda_4=9.2307;$$

Перевірити вірність розв’язку можна за допомогою математичних пакетів:

Вхідні дані:

eigenvalues $\begin{pmatrix} 6.3 & 1.07 & 0.99 & 1.2 \\ 1.07 & 4.12 & 1.3 & 0.16 \\ 0.99 & 1.3 & 5.48 & 2.1 \\ 1.2 & 0.16 & 2.1 & 6.06 \end{pmatrix}$

Результати: Точні форми | Покрокове рішення

$\lambda_1 \approx 9.23121$

$\lambda_2 \approx 5.45095$

$\lambda_3 \approx 4.47197$

$\lambda_4 \approx 2.80586$

Відповідні власні вектори: Точні форми | Покрокове рішення

$v_1 \approx (0.902692, 0.464169, 0.958912, 1)$

$v_2 \approx (-1.49147, -0.492973, 0.599807, 1)$

$v_3 \approx (0.687408, -1.32628, -1.04796, 1)$

$v_4 \approx (-0.444932, 1.64598, -1.42075, 1)$

Результати роботи програми

```
_____________________________________________________

Start data:

-matrix of operator:
6.3     1.07    0.99    1.2
1.07    4.12    1.3     0.16
0.99    1.3     5.48    2.1
1.2     0.16    2.1     6.06

-start vector:
0
0
0
1

_____________________________________________________

Find matrix of linear system, wich we must solve, and vector of righ side:

-matrix:
190.148 17.0822 1.2     0
81.6612 5.6428  0.16    0
254.158 25.63   2.1     0
333.376 42.5992 6.06    1

-vector:
-1936.98
-923.648
-2387.28
-2795.23

_____________________________________________________

Search for coefficients of characteristic polynomial:

190.148*x0 + 17.0822*x1 + 1.2*x2 + 0*x3 = -1936.98
81.6612*x0 + 5.6428*x1 + 0.16*x2 + 0*x3 = -923.648
254.158*x0 + 25.63*x1 + 2.1*x2 + 0*x3 = -2387.28
333.376*x0 + 42.5992*x1 + 6.06*x2 + 1*x3 = -2795.23

Solution:
-21.96
169.721
-550.44
631.388

_____________________________________________________

Counting roots...

Root of polynom -one:

 -method bisection :

         #1 iteration      interval=[2;3];

         #2 iteration      interval=[2.5;3];

         #3 iteration      interval=[2.75;3];
```

```
        #4 iteration      interval=[2.75;2.875];
        #5 iteration      interval=[2.75;2.8125];
        #6 iteration      interval=[2.78125;2.8125];
        #7 iteration      interval=[2.79688;2.8125];
        #8 iteration      interval=[2.80469;2.8125];
        #9 iteration      interval=[2.80469;2.80859];
        #10 iteration     interval=[2.80469;2.80664];
        #11 iteration     interval=[2.80566;2.80664];
        #12 iteration     interval=[2.80566;2.80615];
        #13 iteration     interval=[2.80591;2.80615];
        #14 iteration     interval=[2.80603;2.80615];
        #15 iteration     interval=[2.80609;2.80615];
        #16 iteration     interval=[2.80612;2.80615];
        #17 iteration     interval=[2.80614;2.80615];
        #18 iteration     interval=[2.80614;2.80615];
-result of bisection: 2.80615
Root of polynom -two:
 -method bisection :
        #1 iteration      interval=[4;4.5];
        #2 iteration      interval=[4.25;4.5];
        #3 iteration      interval=[4.375;4.5];
        #4 iteration      interval=[4.4375;4.5];
        #5 iteration      interval=[4.46875;4.5];
        #6 iteration      interval=[4.46875;4.48438];
        #7 iteration      interval=[4.46875;4.47656];
        #8 iteration      interval=[4.46875;4.47266];
        #9 iteration      interval=[4.4707;4.47266];
        #10 iteration     interval=[4.47168;4.47266];
        #11 iteration     interval=[4.47168;4.47217];
        #12 iteration     interval=[4.47168;4.47192];
        #13 iteration     interval=[4.47168;4.4718];
        #14 iteration     interval=[4.47174;4.4718];
```

```
        #15 iteration     interval=[4.47177;4.4718];
        #16 iteration     interval=[4.47179;4.4718];
        #17 iteration     interval=[4.47179;4.47179];
-result of bisection: 4.47179
Root of polynom -three:
 -method bisection :
        #1 iteration      interval=[5;5.5];
        #2 iteration      interval=[5.25;5.5];
        #3 iteration      interval=[5.375;5.5];
        #4 iteration      interval=[5.4375;5.5];
        #5 iteration      interval=[5.4375;5.46875];
        #6 iteration      interval=[5.45312;5.46875];
        #7 iteration      interval=[5.45312;5.46094];
        #8 iteration      interval=[5.45312;5.45703];
        #9 iteration      interval=[5.45312;5.45508];
        #10 iteration     interval=[5.4541;5.45508];
        #11 iteration     interval=[5.45459;5.45508];
        #12 iteration     interval=[5.45459;5.45483];
        #13 iteration     interval=[5.45459;5.45471];
        #14 iteration     interval=[5.45465;5.45471];
        #15 iteration     interval=[5.45465;5.45468];
        #16 iteration     interval=[5.45465;5.45467];
        #17 iteration     interval=[5.45465;5.45466];
-result of bisection: 5.45465
Root of polynom -four:
 -method bisection :
        #1 iteration      interval=[9;12];
        #2 iteration      interval=[9;10.5];
        #3 iteration      interval=[9;9.75];
        #4 iteration      interval=[9;9.375];
        #5 iteration      interval=[9.1875;9.375];
        #6 iteration      interval=[9.1875;9.28125];
```

```
          #7 iteration      interval=[9.1875;9.23438];
          #8 iteration      interval=[9.21094;9.23438];
          #9 iteration      interval=[9.22266;9.23438];
          #10 iteration     interval=[9.22852;9.23438];
          #11 iteration     interval=[9.22852;9.23145];
          #12 iteration     interval=[9.22998;9.23145];
          #13 iteration     interval=[9.22998;9.23071];
          #14 iteration     interval=[9.23035;9.23071];
          #15 iteration     interval=[9.23053;9.23071];
          #16 iteration     interval=[9.23062;9.23071];
          #17 iteration     interval=[9.23067;9.23071];
          #18 iteration     interval=[9.23069;9.23071];
          #19 iteration     interval=[9.2307;9.23071];
          #20 iteration     interval=[9.2307;9.23071];
-result of bisection: 9.2307
```

Код програми

```
#include <iostream>
#include <string>
#include <cmath>
#include <stdio.h>
#include <windows.h>
#include <conio.h>

using namespace std;

#define type long double
#define line
cout<<endl<<"_____________________________________________________"<<endl;



// method A.N. Kirilov block code -start

void output_vector(type* vector,int size)
{
	for (int i=0;i<size;i++)
	{
		cout<<vector[i]<<endl;
	}
}

void output_matrix(type** matrix, int size)
{
	for (int i = 0; i < size; i++)
	{
		for (int j = 0; j < size; j++)
		{
			cout << matrix[i][j] << "\t";
		}
		cout << endl;
	}

}

void null_vector(type* vector, int size)
{
		for (int i = 0; i < size; i++)
		{
			vector[i] = 0;
		}
}

void null_matrix(type**  matrix, int size)
{
	for (int i = 0; i < size; i++)
	{
		for (int j = 0; j < size; j++)
		{
			matrix[i][j] = 0;
		}
	}
}
```

```
type* copy_vector(type* vector_in, type* vector_out, int size)
{
	for (int i = 0; i < size; i++)
	{
		vector_out[i] = vector_in[i];
	}

	return vector_out;
}

type** copy_matrix(type** matrix_in, type** matrix_out, int size)
{
	for (int i = 0; i < size; i++)
	{
		for (int j = 0; j < size; j++)
		{
			matrix_out[i][j] = matrix_in[i][j];
		}
	}

	return matrix_out;
}

type* multiplication_matrix_on_vector(type** matrix,type* vector,type*
result_vector, int size)
{
	null_vector(result_vector, size);

	for (int i = 0; i < size; i++)
	{
		for (int j = 0; j < size; j++)
		{
			result_vector[i]+= matrix[i][j] * vector[j];
		}
	}

	return result_vector;
}

type** finding_matrix_which_we_must_solve(type** operator_matrix, type*
start_vector,type** result_matrix,type* right_vector_of_solved_system,int
size)
{
	type* temp = new type[size];

	null_vector(temp, size);


	for (int i= size-1;i>=0;i--)
	{
		multiplication_matrix_on_vector(operator_matrix, start_vector,
temp, size);

		for (int j = 0; j < size; j++)
		{
			result_matrix[j][i] = start_vector[j];
		}

		copy_vector(temp, start_vector, size);

	}

	copy_vector(temp, right_vector_of_solved_system, size);
```

```
	for (int i = 0; i < size; i++)
	{
		right_vector_of_solved_system[i] = -
right_vector_of_solved_system[i];
	}

	null_vector(start_vector, size);  start_vector[0] = 1;

	delete[] temp;

	return result_matrix;
}

// method A.N. Kirilov block code -end


// method solving SLAE block code -start

void system_output(type **matrix, type *vector, int size)
{
	cout << endl;

	for (int i = 0; i < size; i++)
	{
		for (int j = 0; j < size; j++)
		{
			cout << matrix[i][j] << "*x" << j;
			if (j < size - 1)
			{
				cout << " + ";
			}
		}
		cout << " = " << vector[i] << endl;
	}

}

type * method_Gause(type **matrix, type *vector, int size)
{
	type *solve, max;
	int m, index; m = 0;

	const type epsilone = 0.0001;

	solve = new type[size];

	while (m < size)
	{
		// Поиск строки с максимальным a[i][k]
		max = abs(matrix[m][m]);
		index = m;
		for (int i = m + 1; i < size; i++)
		{
			if (abs(matrix[i][m]) > max)
			{
				max = abs(matrix[i][m]);
				index = i;
			}
		}
		// Перестановка строк

		if (max < epsilone)
		{
			// нет ненулевых диагональных элементов
```

```
                  cout << "The solution cannot be obtained, because of the
zero column:";
                  cout << index << "Matrix" << endl;
                  return 0;
            }
            for (int j = 0; j < size; j++)
            {
                  type temp = matrix[m][j];
                  matrix[m][j] = matrix[index][j];
                  matrix[index][j] = temp;
            }
            type temp = vector[m];
            vector[m] = vector[index];
            vector[index] = temp;

            // Нормализация уравнений
            for (int i = m; i < size; i++)
            {
                  type temp = matrix[i][m];
                  if (abs(temp) < epsilone) continue; // для нулевого
коэффициента пропустить
                  for (int j = 0; j < size; j++)
                        matrix[i][j] = matrix[i][j] / temp;
                  vector[i] = vector[i] / temp;
                  if (i == m)  continue; // уравнение не вычитать само из себя
                  for (int j = 0; j < size; j++)
                        matrix[i][j] = matrix[i][j] - matrix[m][j];
                  vector[i] = vector[i] - vector[m];
            }
            m++;
      }
      // обратная подстановка
      for (m = size - 1; m >= 0; m--)
      {
            solve[m] = vector[m];
            for (int i = 0; i < m; i++)
                  vector[i] = vector[i] - matrix[i][m] * solve[m];
      }
      return solve;
}

// method solving SLAE block code -end


// method bisection for finding root of characteristic polynom block code -
start

type function_(type x)
{
      type result = pow(x, 4) - (21.9633)*pow(x, 3) + 169.7780*pow(x, 2) + -
(550.726)*x + 631.8215;
      return result;
}

type bisection_method(type a, type b, type epsilone)
{
      cout << endl << " -method bisection :" << endl;

      int iteration = 1;

      type start = a;
      type 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;
}

// method bisection for finding root of characteristic polynom block code -
end

int main()
{
	//data -start

	int size=4;

	type start_first;
	type end_first;

	//data -end

	//create block matrix -start

	type** matrix_of_operator = new type*[size];

	for (int i = 0; i < size; i++)
	{
		matrix_of_operator[i] = new type[size];
	}

	type* start_vector = new type[size];

	type** system_must_solving = new type*[size];

	for (int i = 0; i < size; i++)
	{
		system_must_solving[i] = new type[size];
	}

	type* right_vector_of_solved_system = new type[size];

	type* coefitient_polynom= new type[size];

	//create block matrix -end

	//intialization block matrix operator -start

	matrix_of_operator[0][0] = 6.3;		matrix_of_operator[0][1] = 1.07;
matrix_of_operator[0][2] = 0.99;	matrix_of_operator[0][3] = 1.20;
start_vector[0] = 0;

	matrix_of_operator[1][0] = 1.07;	matrix_of_operator[1][1] = 4.12;
matrix_of_operator[1][2] = 1.30;	matrix_of_operator[1][3] = 0.16;
start_vector[1] = 0;
```

```
      matrix_of_operator[2][0] = 0.99;     matrix_of_operator[2][1] = 1.30;
matrix_of_operator[2][2] = 5.48;   matrix_of_operator[2][3] = 2.10;
start_vector[2] = 0;

      matrix_of_operator[3][0] = 1.20;     matrix_of_operator[3][1] = 0.16;
matrix_of_operator[3][2] = 2.10;   matrix_of_operator[3][3] = 6.06;
start_vector[3] = 1;

      //intialization block matrix operator -end



      //program block-start

      line

      cout << endl << "Start data: " << endl;

      cout << endl << "-matrix of operator:" << endl;
      output_matrix(matrix_of_operator, size);

      cout << endl << "-start vector:" << endl;
      output_vector(start_vector, size);

      line

      cout << endl << "Find matrix of linear system, wich we must solve, and
vector of righ side:" << endl;

      cout << endl << "-matrix:" << endl;
      output_matrix(finding_matrix_which_we_must_solve(matrix_of_operator,
start_vector, system_must_solving,right_vector_of_solved_system,size),size);

      cout << endl << "-vector:" << endl;
      output_vector(right_vector_of_solved_system, size);

      line

      cout << endl << "Search for coefficients of characteristic polynomial:"
<< endl;

      type **matrix, *vector, *solution;

      matrix = new type*[size];
      vector = new type[size];

      for (int i = 0; i < size; i++)
      {
            matrix[i] = new type[size];

            for (int j = 0; j < size; j++)
            {
                  matrix[i][j] = system_must_solving[i][j];
            }
      }
      for (int i = 0; i < size; i++)
      {
            vector[i] = right_vector_of_solved_system[i];
      }

      system_output(matrix, vector, size);

      solution = method_Gause(matrix, vector, size);
```

```
	cout << endl << "Solution:" << endl;

	for (int i = 0; i < size; i++)
	{
		cout << solution[i] << endl;
	}

	copy_vector(solution, coefitient_polynom,size);

	line

	{
	cout << endl << "Counting roots..." << endl;

	cout << endl << "Root of polynom -one:" << endl;    start_first = 2;
end_first = 4;

	cout << endl << "-result of bisection: " <<
bisection_method(start_first, end_first, 0.00001) << endl;



	cout << endl << "Root of polynom -two:" << endl;    start_first = 4;
end_first = 5;

	cout << endl << "-result of bisection: " <<
bisection_method(start_first, end_first, 0.00001) << endl;



	cout << endl << "Root of polynom -three:" << endl;  start_first = 5;
end_first = 6;

	cout << endl << "-result of bisection: " <<
bisection_method(start_first, end_first, 0.00001) << endl;


	cout << endl << "Root of polynom -four:" << endl;   start_first = 6;
end_first = 12;

	cout << endl << "-result of bisection: " <<
bisection_method(start_first, end_first, 0.00001) << endl;

	}

	line

	//program block-end

	//delete block matrix -start

	delete[] coefitient_polynom;

	delete[] right_vector_of_solved_system;

	for (int i = 0; i < size; i++)
	{
		delete system_must_solving[i] ;
	}

	delete[] system_must_solving;

	delete[] start_vector;
```

```
	for (int i = 0; i < size; i++)
	{
		delete matrix_of_operator[i];
	}

	delete[] matrix_of_operator;
	//delete block matrix -end

	return 0;
}
```