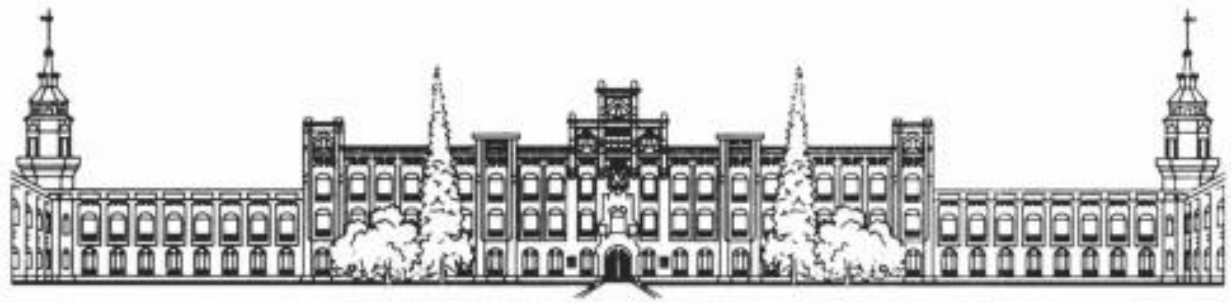

Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря
Сікорського» Інститут прикладного системного аналізу

**Лабораторна робота № 3**
з курсу «Чисельні методи 1»
з теми «Ітераційні методи розв’язання СЛАР»
Варіант № 16

Виконав студент 2 курсу групи КА-91
Панченко Єгор Станіславович
перевірила старший викладач
Хоменко Ольга Володимирівна

Київ-2021

**Завдання: розв'язати СЛАР**

$$\begin{cases} 2.923 \cdot x_1 + 0.220 \cdot x_2 + 0.159 \cdot x_3 + 0.328 \cdot x_4 = 0.605, \\ 0.363 \cdot x_1 + 4.123 \cdot x_2 + 0.268 \cdot x_3 0.327 \cdot x_4 = 0.496, \\ 0.169 \cdot x_1 + 0.271 \cdot x_2 + 3.906 \cdot x_3 + 0.295 \cdot x_4 = 0.590, \\ 0.241 \cdot x_1 + 0.319 \cdot x_2 + 0.257 \cdot x_3 + 3.862 \cdot x_4 = 0.896, \end{cases}$$

**методом Якобі.**

1. **Текст програм**

```
#include <vector>

using namespace std;

namespace Laba3 {

    const vector<vector<double> > A = {{2.923, 0.220, 0.159, 0.328},
                                       {0.363, 4.123, 0.268, 0.327},
                                       {0.169, 0.271, 3.906, 0.295},
                                       {0.241, 0.319, 0.257, 3.862}};
    const vector<double> B = {0.605, 0.496, 0.590, 0.896};
    double eps = 1e-5;
}

using namespace Laba3;

bool DiagonalDominant(vector< vector < double > > A) {
    for (int i = 0; i < A.size(); i++) {
        for (int j = 0; j < A.size(); j++) {
            if (i == j) {
                continue;
            }
            if (A[i][j] >= A[i][i]) {
                return false;
            }
        }
    }

    return true;
}

void PrintMat(vector < vector < double > > A) {
    for (int i = 0; i < A.size(); i++) {
        for (auto j : A[i]) {
            printf("%4.5lf ", j);
        }
        printf("\n");
    }
}

void PrintRow(vector < double > B) {
    for (auto i : B) {
        printf("%4.5lf ", i);
    }
    printf("\n");
}
```

```
void SolveByYacob() {
    bool dom = DiagonalDominant(Laba3::A);
    printf("Diagonal dominant: %s\n", dom ? "true" : "false");
    auto A_it = Laba3::A;
    auto B_it = Laba3::B;
    for (int i = 0; i < A_it.size(); i++) {
        for (int j = 0; j < A_it.size(); j++) {
            if (i == j) {
                A_it[i][j] = 0;
                continue;
            }
            A_it[i][j] = -Laba3::A[i][j] / Laba3::A[i][i];
        }
        B_it[i] /= Laba3::A[i][i];
    }
    printf("\nMatrix B:\n");
    PrintMat(A_it);
    printf("\nColumn c:\n");
    for (int i = 0; i < B_it.size(); i++) {
        printf("c[%d] = %10.5lf\n", i, B_it[i]);
        //PrintRow(B_it);
    }

    vector < vector < double > > x(0);
    bool need_random = true;
    if (need_random) {
        srand(time(0));
        vector < double > r(B_it.size());
        for (auto &i : r) {
            i = (double)rand() / rand();
        }
        x.push_back(r);
    } else {
        x.push_back(B_it);
    }

    for (;;) {
        auto x_it = Multy(A_it, x.back());
        for (int i = 0; i < x_it.size(); i++) {
            x_it[i] += B_it[i];
        }


        double maxdiff = 0;
        for (int i = 0; i < x_it.size(); i++) {
            maxdiff = max(maxdiff, abs(x_it[i] - x.back()[i]));
        }

        x.push_back(x_it);

        if (maxdiff <= Laba3::eps) {
            break;
        }
    }
    printf("\nSolution:\n");
    for (int i = 0; i < x.back().size(); i++) {
        printf("x[%d] = %10.5lf\n", i, x.back()[i]);
```

```
        //PrintRow(x.back());
    }

    auto diff = Substract(Laba3::B, Multy(Laba3::A, x.back()));

    printf("\ndiff is:\n");
    for (auto j : diff) {
        printf("%10.5lf\n", j);
    }
}
```

## 2. Результат роботи програм

| № ітерації | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $\|x^{(k)} - x^{(k-1)}\|$ |
|---|---|---|---|---|---|
| 0 | 1.47970 | 0.04804 | 0.43595 | 0.73210 | - |
| 1 | 0.09750 | -0.09638 | 0.02840 | 0.10669 | 1.382 |
| 2 | 0.20072 | 0.10141 | 0.14546 | 0.23199 | .19779 |
| 3 | 0.16540 | 0.07477 | 0.11781 | 0.20142 | .03532 |
| 4 | 0.17234 | 0.08211 | 0.12349 | 0.20767 | .00734 |
| 5 | 0.17078 | 0.08063 | 0.12221 | 0.20625 | .00156 |
| 6 | 0.17112 | 0.08096 | 0.12249 | 0.20655 | .00034 |
| 7 | 0.17104 | 0.08089 | 0.12243 | 0.20649 | .00008 |
| 8 | 0.17106 | 0.08091 | 0.12244 | 0.20650 | .00002 |
| допоміжні | 1.3822 | .144 | .408 | .625 | |
| обчислення | .10322 | .19779 | .11706 | .1253 | |
| | .03532 | .02664 | .02765 | .03057 | |
| | .00694 | .00734 | .00568 | .00625 | |
| | .00156 | .00148 | .00128 | .00142 | |
| | .00034 | .00033 | .00028 | .0003 | |
| | .00008 | .00007 | .00006 | .00006 | |
| | .00002 | .00002 | .00001 | .00001 | |

```
C:\Users\panen\CLionProjects\NumericalMethods\cmake-build-debug\NumericalM
Diagonal dominant: true

Matrix B:
0.00000 -0.07527 -0.05440 -0.11221
-0.08804 0.00000 -0.06500 -0.07931
-0.04327 -0.06938 0.00000 -0.07552
-0.06240 -0.08260 -0.06655 0.00000

Column c:
c[0] =    0.20698
c[1] =    0.12030
c[2] =    0.15105
c[3] =    0.23200
0-th approximation:
        x[0] =    1.47970
        x[1] =    0.04804
        x[2] =    0.43595
        x[3] =    0.73210
1-th approximation:
        x[0] =    0.09750
        x[1] =   -0.09638
        x[2] =    0.02840
        x[3] =    0.10669
2-th approximation:
        x[0] =    0.20072
        x[1] =    0.10141
        x[2] =    0.14546
        x[3] =    0.23199
3-th approximation:
        x[0] =    0.16540
        x[1] =    0.07477
        x[2] =    0.11781
        x[3] =    0.20142
4-th approximation:
        x[0] =    0.17234
        x[1] =    0.08211
        x[2] =    0.12349
        x[3] =    0.20767
5-th approximation:
        x[0] =    0.17078
        x[1] =    0.08063
        x[2] =    0.12221
        x[3] =    0.20625
6-th approximation:
        x[0] =    0.17112
        x[1] =    0.08096
        x[2] =    0.12249
        x[3] =    0.20655
7-th approximation:
        x[0] =    0.17104
        x[1] =    0.08089
        x[2] =    0.12243
        x[3] =    0.20649
8-th approximation:
        x[0] =    0.17106
        x[1] =    0.08091
        x[2] =    0.12244
        x[3] =    0.20650

Solution:
x[0] =    0.17106
x[1] =    0.08090
x[2] =    0.12244
x[3] =    0.20650

diff is:
   0.00000
   0.00000
   0.00000
   0.00000

Process finished with exit code 0
```

## 3. Висновок

У ході виконання лабораторної роботи було програмно реалізовано метод Якобі для будь-якої СЛАР розмірності $n \times n$, для якої виконується умова діагональної переваги. Перед початком роботи самого методу перевіряється умова діагональної переваги. Програма виводить шуканий вектор-розв'язок. Також вираховується похибка – наскільки відрізняється добуток початкової матриці $A$ і знайденого вектора $\vec{x}$ від початкового вектора $\vec{b}$. За норму брався абсолютний поелементний максимум з різниці двох векторів. Програма виводить послідовність наближень. Початкове наближення можна задати двома способами – вручну або випадковими числами. В програмі реалізований саме випадковий варіант. При декількох запусках розв'язки не змінилися. Кількість наближень коливається в межах 10.