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 №3

з курсу «Чисельні методи»

тема: «МЕТОДИ РОЗВ'ЯЗАННЯ СЛАР»

Виконав: студент 2 курсу

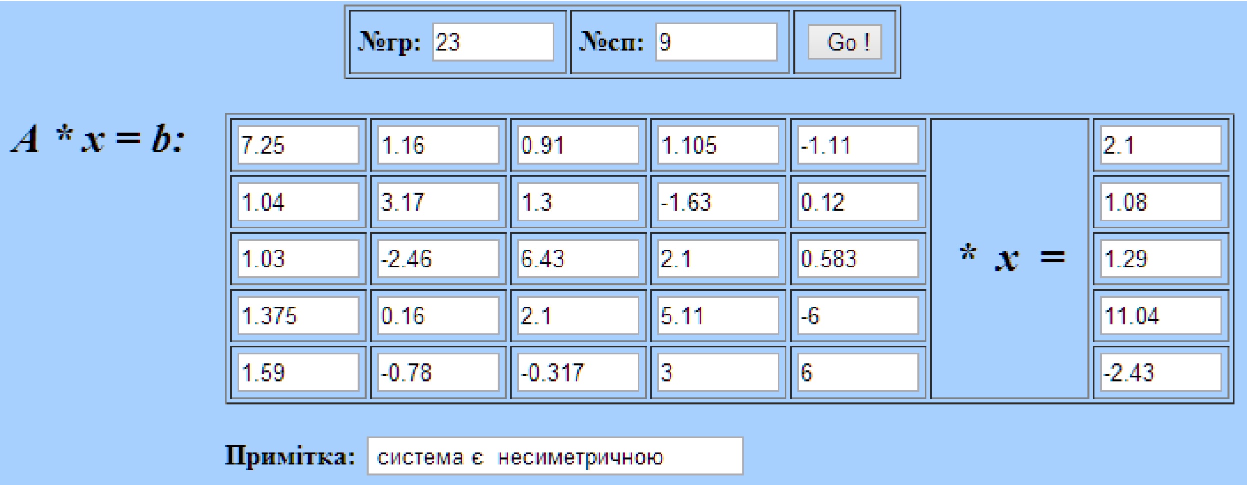
групи КА-23

Деундяк О.В.

Прийняв: Коновалюк М. М.

Київ – 2014р.

Варіант 9



№гр: 23 №сп: 9 Go !

*A * x = b:*

7.25	1.16	0.91	1.105	-1.11		2.1
1.04	3.17	1.3	-1.63	0.12		1.08
1.03	-2.46	6.43	2.1	0.583	* x =	1.29
1.375	0.16	2.1	5.11	-6		11.04
1.59	-0.78	-0.317	3	6		-2.43

Примітка: система є несиметричною

Умова: Знайти розв’язок рівняння Ax=b ітераційним методом.

Допрограмовий етап

Для розв’язку системи я обрав метод Гауса-Зейделя, оскільки дана матриця додатньо визначена.

Текст програми:

Matrix.h

```
#pragma once

#include <vector>
#include <string>
#include <fstream>

using std::vector;
using std::string;
using std::ifstream;

typedef double Item;

class Matrix
{
	vector <vector <Item>> A;
	size_t n;

	vector<Item> SolveL(const vector<Item> &b) const;
	vector<Item> SolveU(const vector<Item> &b) const;
public:
	Matrix(size_t dimension, Item fill);
	Matrix(size_t dimension);
	Matrix(size_t dimension, ifstream &file);

	Item Determinant() const;
	bool DiagDom() const;

	Matrix Inverse();

	Matrix operator*(const Matrix &other) const;
	vector<Item> operator*(const vector<Item> &vec) const;
```

```cpp
        vector<Item> Solve(const vector<Item> &b) const;
        vector<Item> Solve2(const vector<Item> &b) const;

        void Show() const;
        void LU(Matrix &L, Matrix &U) const;
};
```

Matrix.cpp

```cpp
#include "Matrix.h"
#include <iostream>
#include <iomanip>
#include <fstream>
#include <exception>
#include <algorithm>

const double eps = 1E-5;
const long MAXIT = 1000;

using std::cin;
//using std::cout;
using std::endl;
using std::ifstream;

extern std::ofstream cout;

Matrix::Matrix(size_t n_, Item fill) :
        n(n_)
{
        for (size_t i = 1; i <= n; i++)
        {
                vector<Item> T;
                for (size_t j = 1; j <= n; j++)
                        T.push_back(fill);
                A.push_back(T);
        }
}


Matrix::Matrix(size_t n_, ifstream &file) :
        n(n_)
{
        for (size_t i = 1; i <= n; i++)
        {
                vector<Item> T;
                Item t;
                for (size_t j = 1; j <= n; j++)
                {
                        file >> t;
                        T.push_back(t);
                }
                A.push_back(T);
        }
}

Matrix::Matrix(size_t n_) :
        n(n_)
{
        for (size_t i = 1; i <= n; i++)
        {
                vector<Item> T;
                Item t;
                cout << "Enter raw #" << i << ": ";
                for (size_t j = 1; j <= n; j++)
                {
                        cin >> t;
```

```
            T.push_back(t);
        }
        A.push_back(T);
    }
    cout << endl;
}

void Matrix::LU(Matrix &L, Matrix &U) const
{
    U = *this;

    for (size_t i = 0; i < n; i++)
    for (size_t j = i; j < n; j++)
    {
        if (abs(U.A[i][i]) < eps)
            throw std::logic_error("LU decompsition doesn't exist");
        L.A[j][i] = U.A[j][i] / U.A[i][i];
    }

    for (size_t k = 1; k < n; k++)
    {
        for (size_t i = k - 1; i < n; i++)
        for (size_t j = i; j < n; j++)
            L.A[j][i] = U.A[j][i] / U.A[i][i];

        for (size_t i = k; i < n; i++)
        for (size_t j = k - 1; j < n; j++)
            U.A[i][j] = U.A[i][j] - L.A[i][k - 1] * U.A[k - 1][j];
    }
}

void Matrix::Show() const
{
    std::streamsize pr = cout.precision(5);
    cout << std::left;
    cout << "MATRIX " << n << 'x' << n << ": " << endl;
    for (size_t i = 0; i < n; i++)
    {
        for (size_t j = 0; j < n; j++)
        {
            cout << std::setw(10);
            cout << A[i][j];
            cout << ' ';
        }
        cout << endl;
    }
    cout << endl;
    cout.precision(pr);
}

Matrix Matrix::operator*(const Matrix &other) const
{
    Matrix P(n, 0);

    for (size_t row = 0; row < n; row++)
        for (size_t col = 0; col < n; col++)
            for (size_t inner = 0; inner < n; inner++)
                P.A[row][col] += A[row][inner] * other.A[inner][col];

    return P;
}

vector<Item> Matrix::SolveL(const vector<Item> &b) const
{
    vector<Item> r;
    r.reserve(n);
    Item t;
    for (size_t i = 0; i < n; i++)
    {
        t = 0;
        for (size_t j = 0; j < i; j++)
            t += A[i][j] * r[j];
        r.push_back((b[i] - t) / A[i][i]);
    }

    return r;
}

vector<Item> Matrix::SolveU(const vector<Item> &b) const
{
    vector<Item> r;
    r.resize(n);
    Item t;
    for (size_t i = n; i > 0; i--)
    {
        t = 0;
        for (size_t j = n-1; j > i - 1; j--)
            t += A[i - 1][j] * r[j];
        r[i - 1] = (b[i - 1] - t) / A[i - 1][i - 1];
    }

    return r;
}

vector<Item> Matrix::Solve(const vector<Item> &b) const
{
    Matrix L(n, 0), U(n, 0);
    LU(L, U);

    //(L*U).Show();

    return U.SolveU(L.SolveL(b));
}

Item Matrix::Determinant() const
{
    Matrix L(n, 0), U(n, 0);

    Item det = 1;

    LU(L, U);
    for (size_t i = 0; i < n; i++)
        det *= U.A[i][i];

    return det;
}

Matrix Matrix::Inverse()
{
    Matrix m_inv(n, 0);

    vector<Item> b, r;
    b.resize(n);

    for (size_t i = 0; i < n; i++)
    {
        for (size_t j = 0; j < n; j++)
            b[j] = (i == j) ? 1 : 0;
        r = Solve(b);
```

```cpp
        for (size_t j = 0; j < n; j++)
            (m_inv.A)[j][i] = r[j];
    }

    return m_inv;
}

vector<Item> Matrix::operator*(const vector<Item> &vec) const
{
    vector<Item> res;
    for (size_t i = 0; i < n; i++)
    {
        Item r = 0;
        for (size_t j = 0; j < n; j++)
            r += A[i][j] * vec[j];
        res.push_back(r);
    }

    return res;
}

bool Matrix::DiagDom() const
{
    Item s;
    for (size_t i = 0; i < n; i++)
    {
        s = 0;
        for (size_t j = 0; j < n; j++)
            s += (i != j) ? abs(A[i][j]) : 0.0;
        if (abs(A[i][i]) < s)
            return false;
    }

    return true;
}

Item operator*(const vector<Item> &v1, const vector<Item> &v2)
{
    size_t l = std::min(v1.size(), v2.size());
    Item r = 0;
    for (size_t i = 0; i < l; i++)
        r += v1[i] * v2[i];

    return r;
}

vector<Item> operator-(const vector<Item> &v1, const vector<Item> &v2)
{
    size_t l = std::min(v1.size(), v2.size());
    vector<Item> r;
    r.resize(l);
    for (size_t i = 0; i < l; i++)
        r[i] = v1[i] - v2[i];

    return r;
}

vector<Item> operator+(const vector<Item> &v1, const vector<Item> &v2)
{
    size_t l = std::min(v1.size(), v2.size());
    vector<Item> r;
    r.resize(l);
    for (size_t i = 0; i < l; i++)
        r[i] = v1[i] + v2[i];
    return r;
}

void ShowV(const vector<double> &v)
{
    cout << "{";
    for (size_t i = 0; i < v.size(); i++)
        cout << v[i] << " ";
    cout << "}" << endl;
}

vector<Item> Matrix::Solve2(const vector<double> &b) const
{
    //if (!DiagDom())
        //throw std::exception("Matrix is not diagonally dominant");

    Matrix L = *this, U = *this;
    for (size_t i = 0; i < n; i++)
    {
        for (size_t j = i + 1; j < n; j++)
            L.A[i][j] = 0;

    }
    for (size_t i = 0; i < n; i++)
    for (size_t j = 0; j < n; j++)
        U.A[i][j] = (j <= i) ? 0 : -A[i][j];

    vector<Item> x = b, e = b - (*this)*x;;
    Item ep = sqrt(e*e);
    long it = 1;

    while (ep > eps)
    {
        cout << "Iteration " << it << endl << "x=";
        ShowV(x);
        cout << "eps=";
        ShowV(e);
        cout << "|eps|=" << ep << endl;
        x = L.SolveL(U*x + b);
        e = b - (*this)*x;
        ep = sqrt(e * e);
        ++it;
        if (it > MAXIT)
            throw std::exception("Too many iterations");
    }

    cout << "Result in " << it << " iterations:" << endl << "x=";
    ShowV(x);
    cout << "eps=";
    ShowV(e);
    cout << "|eps|=" << ep << endl;

    return x;
}
```

SOLE.h

```cpp
#pragma once
#include "Matrix.h"
#include <memory>

using std::shared_ptr;

class SOLE
{
	shared_ptr<Matrix> A;
	vector<Item> b;
public:
	void start();
};
```

SOLE.cpp

```
#include "SOLE.h"
#include <iostream>
#include <string>
#include <fstream>

const double eps = 1E-5;

using std::cin;
//using std::cout;
using std::endl;
using std::ifstream;

std::ofstream cout("output.txt");

const string INFILE = "input.txt";

void SOLE::start()
{
	ifstream file(INFILE);
	size_t n;
	if (!file.is_open())
		throw std::runtime_error("Can't open file");

	file >> n;
	A = std::make_shared<Matrix>(n, file);
	cout << "Input Matrix" << endl;
	A->Show();

	for (size_t i = 0; i < n; i++)
	{
		Item t;
		file >> t;
		b.push_back(t);
	}

	A->Solve2(b);

}
```

main.cpp

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


using std::cin;
using std::cout;

int main()
{
	SOLE sole;
	try
	{
		sole.start();
	}
	catch (std::exception e)
	{
        cout << e.what();
    }

    cout << std::endl << "DONE";
    cin.get();
    cin.get();
}
```

Результати роботи програми

```
Input Matrix
MATRIX 5x5:
7.25    1.16    0.91    1.105    -1.11
1.04    3.17    1.3     -1.63    0.12
1.03    -2.46   6.43    2.1      0.583
1.375   0.16    2.1     5.11     -6
1.59    -0.78   -0.317  3        6

Iteration 1
x={2.1 1.08 1.29 11.04 -2.43 }
eps={-30.4482 12.0822 -28.2782 -65.7237 -23.0577 }
|eps|=81.9999

Iteration 2
x={-2.09975 6.26926 -0.449797 -0.139207 1.01228 }
eps={11.7376 -16.3734 21.4695 20.6537 -4.44089e-016 }
|eps|=35.9635

Iteration 3
x={-0.480768 0.572983 0.45053 3.27533 -1.81697 }
eps={-1.12515 4.73478 -5.52107 -16.9755 -4.44089e-016 }
|eps|=18.5022

Iteration 4
x={-0.635961 2.11752 0.207658 0.0465252 0.0265156 }
eps={4.04345 -5.16843 5.70573 11.0609 0 }
|eps|=14.0699

Iteration 5
x={-0.0782438 0.304124 0.311909 2.07495 -1.36573 }
eps={-1.77813 3.33788 -3.44802 -8.35345 4.44089e-016 }
|eps|=9.79654

Iteration 6
x={-0.323504 1.43755 0.248583 0.496756 -0.367635 }
eps={1.59464 -2.60991 2.73233 5.98855 0 }
|eps|=7.25829
```

```
Iteration 7
x={-0.103553 0.542071 0.295692 1.61818 -1.10056 }
eps={-1.05683 1.85463 -1.92769 -4.39752 -4.44089e-016 }
|eps|=5.25459

Iteration 8
x={-0.249322 1.17495 0.261373 0.791117 -0.567935 }
eps={0.802202 -1.36741 1.42631 3.19572 4.44089e-016 }
|eps|=3.84192

Iteration 9
x={-0.138674 0.70729 0.286552 1.39103 -0.956677 }
eps={-0.574828 0.991769 -1.03317 -2.33245 0 }
|eps|=2.79675

Iteration 10
x={-0.21796 1.04616 0.268219 0.952835 -0.673486 }
eps={0.422133 -0.724401 0.7551 1.69915 4.44089e-016 }
|eps|=2.03966

Iteration 11
x={-0.159735 0.798542 0.281591 1.27194 -0.879952 }
eps={-0.306717 0.527533 -0.54975 -1.2388 0 }
|eps|=1.48634

Iteration 12
x={-0.202041 0.978836 0.271847 1.03926 -0.729476 }
eps={0.22387 -0.384664 0.400907 0.902856 0 }
|eps|=1.08349

Iteration 13
x={-0.171162 0.84736 0.27895 1.20883 -0.839162 }
eps={-0.163082 0.280332 -0.292156 -0.658115 4.44089e-016 }
|eps|=0.789717

Iteration 14
x={-0.193656 0.943173 0.273773 1.08522 -0.759214 }
eps={0.118899 -0.204347 0.21297 0.479686 -4.44089e-016 }
|eps|=0.57563

Iteration 15
x={-0.177257 0.87333 0.277547 1.17532 -0.817488 }
eps={-0.0866552 0.148942 -0.155226 -0.349643 -4.44089e-016 }
|eps|=0.419569

Iteration 16
```

```
x={-0.189209 0.924236 0.274796 1.10964 -0.775013 }
eps={0.0631653 -0.108565 0.113145 0.254851 -4.44089e-016 }
|eps|=0.305822

Iteration 17
x={-0.180497 0.88713 0.276801 1.15751 -0.805973 }
eps={-0.0460398 0.0791315 -0.0824703 -0.18576 4.44089e-016 }
|eps|=0.222911

Iteration 18
x={-0.186847 0.914177 0.27534 1.12262 -0.783406 }
eps={0.0335584 -0.0576786 0.0601122 0.135399 0 }
|eps|=0.162479

Iteration 19
x={-0.182218 0.894463 0.276405 1.14805 -0.799855 }
eps={-0.0244604 0.0420415 -0.0438153 -0.0986914 4.44089e-016 }
|eps|=0.11843

Iteration 20
x={-0.185592 0.908832 0.275629 1.12952 -0.787866 }
eps={0.0178291 -0.0306438 0.0319367 0.0719355 -4.44089e-016 }
|eps|=0.0863226

Iteration 21
x={-0.183133 0.898358 0.276194 1.14303 -0.796604 }
eps={-0.0129955 0.022336 -0.0232785 -0.0524333 -4.44089e-016 }
|eps|=0.0629199

Iteration 22
x={-0.184925 0.905992 0.275782 1.13318 -0.790235 }
eps={0.00947231 -0.0162806 0.0169675 0.0382183 0 }
|eps|=0.0458619

Iteration 23
x={-0.183619 0.900428 0.276083 1.14036 -0.794878 }
eps={-0.0069043 0.0118668 -0.0123675 -0.027857 -4.44089e-016 }
|eps|=0.0334284

Iteration 24
x={-0.184571 0.904484 0.275863 1.13513 -0.791493 }
eps={0.0050325 -0.00864963 0.00901459 0.0203048 4.44089e-016 }
|eps|=0.0243658

Iteration 25
x={-0.183877 0.901528 0.276023 1.13894 -0.79396 }
```

```
eps={-0.00366815 0.00630466 -0.00657067 -0.0148 4.44089e-016 }
|eps|=0.01776

Iteration 26
x={-0.184383 0.903682 0.275907 1.13616 -0.792162 }
eps={0.00267369 -0.00459542 0.00478932 0.0107876 4.44089e-016 }
|eps|=0.0129452

Iteration 27
x={-0.184014 0.902112 0.275992 1.13819 -0.793473 }
eps={-0.00194884 0.00334957 -0.0034909 -0.00786304 -4.44089e-016 }
|eps|=0.00943564

Iteration 28
x={-0.184283 0.903257 0.27593 1.13671 -0.792517 }
eps={0.00142049 -0.00244148 0.00254449 0.00573132 4.44089e-016 }
|eps|=0.00687758

Iteration 29
x={-0.184087 0.902422 0.275975 1.13778 -0.793214 }
eps={-0.00103539 0.00177958 -0.00185466 -0.00417752 -4.44089e-016 }
|eps|=0.00501302

Iteration 30
x={-0.18423 0.90303 0.275942 1.137 -0.792706 }
eps={0.000754687 -0.00129712 0.00135185 0.00304496 0 }
|eps|=0.00365396

Iteration 31
x={-0.184126 0.902587 0.275966 1.13757 -0.793076 }
eps={-0.000550087 0.000945464 -0.000985356 -0.00221945 4.44089e-016 }
|eps|=0.00266334

Iteration 32
x={-0.184202 0.90291 0.275949 1.13716 -0.792807 }
eps={0.000400954 -0.000689142 0.000718219 0.00161774 4.44089e-016 }
|eps|=0.00194129

Iteration 33
x={-0.184146 0.902675 0.275961 1.13746 -0.793003 }
eps={-0.000292253 0.000502311 -0.000523505 -0.00117916 0 }
|eps|=0.001415

Iteration 34
x={-0.184187 0.902846 0.275952 1.13724 -0.79286 }
eps={0.000213021 -0.000366131 0.000381579 0.000859484 4.44089e-016 }
```

```
|eps|=0.00103138

Iteration 35
x={-0.184157 0.902721 0.275959 1.1374 -0.792964 }
eps={-0.00015527 0.00026687 -0.000278131 -0.000626472 0 }
|eps|=0.000751766

Iteration 36
x={-0.184179 0.902812 0.275954 1.13728 -0.792888 }
eps={0.000113175 -0.00019452 0.000202727 0.000456631 0 }
|eps|=0.000547957

Iteration 37
x={-0.184163 0.902746 0.275957 1.13737 -0.792944 }
eps={-8.24925e-005 0.000141784 -0.000147767 -0.000332835 4.44089e-016 }
|eps|=0.000399402

Iteration 38
x={-0.184174 0.902794 0.275955 1.1373 -0.792903 }
eps={6.01282e-005 -0.000103346 0.000107706 0.000242601 4.44089e-016 }
|eps|=0.000291122

Iteration 39
x={-0.184166 0.902759 0.275957 1.13735 -0.792933 }
eps={-4.3827e-005 7.53279e-005 -7.85063e-005 -0.000176831 -4.44089e-016 }
|eps|=0.000212197

Iteration 40
x={-0.184172 0.902785 0.275955 1.13732 -0.792911 }
eps={3.19452e-005 -5.4906e-005 5.72227e-005 0.000128891 0 }
|eps|=0.000154669

Iteration 41
x={-0.184168 0.902766 0.275956 1.13734 -0.792927 }
eps={-2.32847e-005 4.00206e-005 -4.17092e-005 -9.39475e-005 -4.44089e-016 }
|eps|=0.000112737

Iteration 42
x={-0.184171 0.90278 0.275956 1.13732 -0.792915 }
eps={1.6972e-005 -2.91708e-005 3.04016e-005 6.84777e-005 -4.44089e-016 }
|eps|=8.21732e-005

Iteration 43
x={-0.184169 0.90277 0.275956 1.13734 -0.792924 }
eps={-1.23708e-005 2.12624e-005 -2.21595e-005 -4.99129e-005 -4.44089e-016 }
|eps|=5.98955e-005
```

```
Iteration 44
x={-0.18417 0.902777 0.275956 1.13733 -0.792918 }
eps={9.01699e-006 -1.5498e-005 1.61519e-005 3.63812e-005 4.44089e-016 }
|eps|=4.36574e-005

Iteration 45
x={-0.184169 0.902772 0.275956 1.13733 -0.792922 }
eps={-6.57242e-006 1.12964e-005 -1.1773e-005 -2.6518e-005 0 }
|eps|=3.18216e-005

Iteration 46
x={-0.18417 0.902776 0.275956 1.13733 -0.792919 }
eps={4.79059e-006 -8.23386e-006 8.58127e-006 1.93288e-005 0 }
|eps|=2.31945e-005

Iteration 47
x={-0.184169 0.902773 0.275956 1.13733 -0.792921 }
eps={-3.49183e-006 6.0016e-006 -6.25483e-006 -1.40886e-005 4.44089e-016 }
|eps|=1.69063e-005

Iteration 48
x={-0.18417 0.902775 0.275956 1.13733 -0.792919 }
eps={2.54517e-006 -4.37453e-006 4.5591e-006 1.02691e-005 0 }
|eps|=1.23229e-005

Result in 49 iterations:
x={-0.184169 0.902773 0.275956 1.13733 -0.792921 }
eps={-1.85516e-006 3.18856e-006 -3.3231e-006 -7.48508e-006 -4.44089e-016 }
|eps|=8.98209e-006
```

Висновки:

Метод Гауса-Зейделя – добре працює для розв’язку СЛАР і дає достатньо непогано швидкість збіжності.