МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Лабораторная работа №1

По курсу «Вычислительные методы алгебры»

Метод Гаусса (схема единственного деления)

Вариант №5

Работу выполнил:
студент 3 курса 7 группы
Шатерник Артём
Преподаватель:
Будник А. М.

Минск 2023

1. Постановка задачи.

Найти решение системы линейных алгебраических уравнений $Ax = b$ с расширенной матрицей вида

A					b
0.5757	-0.0758	0.0152	0.0303	0.1061	3.5148
0.0788	0.9014	0.0000	-0.0606	0.0606	3.8542
0.0455	0.0000	0.7242	-0.2121	0.1212	-4.9056
-0.0909	0.1909	0.0000	0.7121	-0.0303	2.3240
0.3788	0.0000	0.1364	0.0152	0.8484	0.1818

применяя метод Гаусса (схема единственного деления).
Найти обратную матрицу A^{-1}, число обусловленности, вектора невязок для решения первоначальной матрицы и для вычисления обратной матрицы.

2. Алгоритм решения.

Пусть дана система:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2 \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n \end{cases}$$

Прямой ход.

Приводим её к верхнетреугольному виду с помощью следующего алгоритма из $i = \overline{1, n}$ итераций:

1. Выбираем элемент $a_{ii} \neq 0$ в качестве ведущего элемента и делим i-ую строку на a_{ii}.
2. Исключаем x_i из всех остальных уравнений, вычитая из них i-ое уравнение, умноженное на a_{ji}, где j – номер соответствующего уравнения.

Получим систему с верхнетреугольным видом

$$\begin{cases} x_1 + c_{12}x_2 + \dots + c_{1n}x_n = q_1 \\ \qquad\qquad x_2 + \dots + c_{2n}x_n = q_2 \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ \qquad\qquad\qquad\qquad\qquad x_n = q_n \end{cases}$$

Обратный ход.
Получаем $\boldsymbol{x_i}$ по формулам:
$\boldsymbol{x_n = q_n}$
$\boldsymbol{x_i = q_i - \sum_{j=i+1}^{n} c_{ij}\, x_j}\,,\ \ i = \overline{n-1, 1}\,.$

Определитель матрицы вычисляется по формуле:
$\det A = a_{11}^1 * \ a_{22}^2 * \ \cdots \ * \ a_{nn}^n$
т.е. он равен произведению ведущих элементов при каждой итерации.

Нахождение обратной матрицы.
Столбцы x^i обратной матрицы вычисляются из системы $Ax^i = \delta^i$, где $\delta^i = (\delta_{1i}\,, \delta_{2i}\,, \cdots\,, \delta_{ni})^T$ и δ_{ij} – символ Кронекера.

Невязки.
Невязка для вектора решений: $\boldsymbol{r = Ax - b}$ **.**
Невязка для обратной матрицы: $\boldsymbol{R = A^{-1}A - E}$ **.**

Число обусловленности.
$\boldsymbol{\nu = \|A\| \cdot \|A^{-1}\|}$
Использовалась первая матричная норма: $\max_i \sum_{j=1}^{n} |a_{ij}|$

3. Листинг программы.

Функция для метода Гаусса.

```
template <typename T>
std::vector < std::vector <T>> gaussian_elimination(int size,
	std::vector <std::vector <T>> a_matrix,
	std::vector <std::vector <T>> b_vector) {
	// Метод Гаусса
	std::vector <std::vector <T>> x_result(size, std::vector <T>(1));
	long double det_a = 1;
	// Прямой ход
	for (int i = 0; i < size; i++) {
		long double leading_elem = a_matrix[i][i];
		// Проверить что ведущий элемент не близок к нулю
		for (int k = i + 1; k < size; k++) {
			if (abs(leading_elem) < 0.00001) {
				a_matrix[i][i] = 0;
				std::swap(a_matrix[i], a_matrix[k]);
				std::swap(b_vector[i], b_vector[k]);
				leading_elem = a_matrix[i][i];
			}
			else {
				break;
			}
		}
	// Вычисляем определитель
		det_a *= leading_elem;
		b_vector[i] = b_vector[i] * (1 / leading_elem);
```

```
            a_matrix[i] = a_matrix[i] * (1 / leading_elem);
            for (int j = i + 1; j < size; j++) {
                b_vector[j] = b_vector[j] - a_matrix[j][i] * b_vector[i];
                a_matrix[j] = a_matrix[j] - a_matrix[j][i] * a_matrix[i];
                a_matrix[j][i] = 0;
            }
        }
        // Обратный ход
        for (int i = size - 1; i >= 0; i--) {
            x_result[i][0] = b_vector[i][0];
            for (int j = size - 1; j > i; j--) {
                x_result[i][0] -= x_result[j][0] * a_matrix[i][j];
            }
        }
        return x_result;
}
// Первая матричная норма
template <typename T>
T first_matrix_norm(std::vector <std::vector <T>> matrix) {
        int size = matrix.size();
        T sum = 0;
        for (int i = 0; i < size; i++) {
            for (int j = 0; j < size; j++) {
                matrix[i][j] = abs(matrix[i][j]);
            }
        }
        for (int i = 0; i < size; i++) {
            auto temp = matrix[i];

            sum += *std::max_element(begin(temp), end(temp));
        }
        return sum;
}
// Произведение матриц
template <typename T>
std::vector <std::vector <T>> matrix_product(std::vector <std::vector <T>> a_matrix,
        std::vector <std::vector <T>> b_matrix) {
        int res_rows = a_matrix.size();
        int res_columns = b_matrix[0].size();
        std::vector <std::vector <T>> result(res_rows, std::vector <T>(res_columns));
        if (a_matrix[0].size() == b_matrix.size()) {
            for (int i = 0; i < res_rows; i++) {
                for (int j = 0; j < res_columns; j++) {
                    for (int k = 0; k < a_matrix[0].size(); k++) {
                        result[i][j] += a_matrix[i][k] * b_matrix[k][j];
                    }
                }
            }
        }
        else {
            throw "incorret matrix size";
        }
        return result;
}

int main() {
        // Ввод данных
        int size = 5;
        std::vector <std::vector <long double>> x_result(size, std::vector <long double>(1));
```

```cpp
	std::vector <std::vector <long double>> a_matrix(size, std::vector <long double> (size));
	std::vector <std::vector <long double>> b_vector(size, std::vector <long double>(1));
	std::ifstream input("input.txt");
	for (int i = 0; i < size; i++) {
		for (int j = 0; j < size; j++) {
			input >> a_matrix[i][j];
		}
	}
	for (int i = 0; i < size; i++) {
		input >> b_vector[i][0];
	}
	x_result = gaussian_elimination(size, a_matrix, b_vector);
	// Обратная матрица
	std::vector <std::vector <long double>> inv_matrix(size, std::vector <long double>(size));
	for (int i = 0; i < size; i++) {
		std::vector <std::vector <long double>> e_vec(size, std::vector <long double>(1));
		e_vec[i][0] = 1;
		std::vector <std::vector <long double>> inv_column = gaussian_elimination(size, a_matrix, e_vec);
		for (int j = 0; j < size; j++) {
			inv_matrix[j][i] = inv_column[j][0];
		}
	}
	// Вычисление невязок
	std::vector <std::vector <long double>> r = matrix_product(a_matrix, x_result) - b_vector;
	std::vector <std::vector <long double>> r_inv = matrix_product(inv_matrix, a_matrix);
	for (int i = 0; i < size; i++) {
		r_inv[i][i] -= 1;
	}
// Число обусловленности
	long double u = first_matrix_norm(a_matrix) * first_matrix_norm(inv_matrix);
	return 0;
}
```

4. Результат и его анализ.

A в верхнетреугольном виде:

1	-0.1317	0.0264	0.0526	0.1843	|6.1053
0	1	-0.0023	-0.071	0.0505	|3.6995
0	0	1	-0.2961	0.1556	|-7.1998
0	0	0	1	-0.0315	|3.0629
0	0	0	0	1	|-2.0007

x = (7.0012 3.9999 -6.0003 2.9999 -2.0007)

det A = 0.209991

A^-1 =

1.8679	0.1692	0.0077	-0.0575	-0.2488
-0.0907	1.0803	0.014	0.1013	-0.0642
0.0917	-0.0802	1.4153	0.4149	-0.1931
0.2265	-0.2705	-0.0126	1.367	0.0416
-0.8528	-0.0578	-0.2307	-0.0655	1.3201

Невязка r = Ax - b:

(-4.44089e-16 -8.88178e-16 -8.88178e-16 0 -2.498e-16)

Норма невязки r = 2.47025e-15

Невязка r_inv = (A^-1)A - E:

0	-3.64292e-17	0	2.12504e-17	2.77556e-17
-6.93889e-18	0	-1.73472e-18	-5.42101e-19	0
0	0	-2.22045e-16	-3.72966e-17	0
-6.93889e-18	0	8.67362e-19	-1.11022e-16	0
0	1.56125e-17	2.77556e-17	3.46945e-18	2.22045e-16

Норма невязки r_inv = 5.9848e-16

Число обусловленности:
u = 26.52

Нормы невязок для решения системы и обратной матрицы малы, число обусловленности также небольшое. Можно сказать что полученное решение близко к реальному решению.