ОТЧЁТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

Численное решение систем линейных алгебраических уравнений (Вариант 8)

Выполнил студент 3 курса ПМиИ
Ковшов Максим

Цель занятия: изучение численных методов решения систем линейных алгебраических уравнений, практическое решение систем на ЭВМ.

Задания к работе: Написать, отладить и выполнить программы решения систем линейных алгебраических уравнений, записанных в векторно-матричной форме $A\,x = b$ и приведенных в таблице. В колонке x* приведено точное решение. Решить систему методом Гаусса с выбором главного элемента и методом Зейделя.

Оценить погрешности методов.

№	A				b	x^*
8	2,12	0,42	1,34	0,88	11,172	3,7
	0,42	3,95	1,87	0,43	0,115	−1,5
	1,34	1,87	2,98	0,46	9,009	2,1
	0,88	0,43	0,46	4,44	9,349	1,3

Метод Гаусса с выбором главного элемента:
Результаты вычислений:

```
Метод Гаусса

Треугольный вид матрицы
2.1200000000000001 0.4200000000000000 1.3400000000000001 0.8800000000000000
0.0000000000000000 3.8667924528301887 1.6045283018867926 0.2556603773584906
0.0000000000000000 0.0000000000000000 1.4672186981555577 -0.2023128720601151
0.0000000000000000 0.0000000000000000 0.0000000000000000 4.0299168484142243

Решение методом Гаусса
3.6999999999999997 -1.5000000000000004 2.1000000000000005 1.2999999999999998

Погрешность метода Гаусса
0.0000000000000004
```

Погрешность приближенного решения с точным равно $4 * 10^{-16}$

Метод Зейделя:

Возьмем в качестве начального приближения нулевой вектор. Запустим итерационный процесс. При вычислении нового значения для i-ой переменной используем старые и новые значения для предыдущих переменных.

Результаты вычислений:

```
Метод Зейделя

Метод сошелся на 37 итерации
Предпоследнее приблежение
3.7000000000266935
-1.4999999999782310
2.0999999999758954
1.2999999999950984

Решение методом Зейделя
3.7000000000062903 -1.4999999999948703 2.0999999999943202 1.2999999999988450

Погрешность метода Зейделя
0.0000000000062901
```

Погрешность приближенного решения с точным равно $6 * 10^{-12}$

Метод простых итераций:

Возьмем в качестве начального приближения нулевой вектор и запустим итерационный процесс.

```
Метод Простых итераций

Метод сошелся на 381 итерации
Решение методом Простых итераций
3.7000000000014834 -1.4999999999990050 2.1000000000014865 1.3000000000005854

Погрешность метода Простых итераций
0.0000000000014864
```

Погрешность приближенного решения с точным равно 10^{-12}

Вывод:

Метод Гаусса имеет наименьшую погрешность, что позволяет найти более точно решение. При выполнении условий сходимости метод Зейделя сходится к решению достаточно быстро. Метод простых итераций имеет самую легкую реализацию, однако его скорость является самой низкой.

ПРИЛОЖЕНИЕ

```cpp
#include <iostream>
#include <iomanip>
#include <ios>
#include <vector>
#include <clocale>

template<typename T>
void printTwo(std::vector<std::vector<T>>& array)
{
    for (const auto &i : array)
    {
        for (const auto j : i)
            std::cout << j << " ";
        std::cout << std::endl;
    }
}

template<typename T>
void printOne(std::vector<T>& array)
{
    for (const auto& i : array)
    {
        std::cout << i << " ";
    }
}

void calculationError(std::vector<double>& result, std::vector<double>& answer)
{
    double maxDiff = 0.0;
    for (int i = 0; i < result.size(); i++)
    {
        double diff = abs(result[i] - answer[i]);
        if (diff > maxDiff)
            maxDiff = diff;
    }
    std::cout << std::fixed << std::setprecision(16) << maxDiff << std::endl;
}

```

```cpp
std::vector<double> metodGauss(std::vector<std::vector<double>> matrix, std::vector<double> b)
{
    const int size = matrix.size();
    for (int i = 0; i < size; i++)
    {
        int maxIndex = i;
        double maxEl = std::fabs(matrix[i][i]);
        for (int k = i + 1; k < size; k++)
        {
            const double currenEl = std::fabs(matrix[k][i]);
            if (currenEl > maxEl)
            {
                maxEl = currenEl;
                maxIndex = k;
            }
        }
        if (maxIndex != i)
        {
            std::swap(matrix[i], matrix[maxIndex]);
            std::swap(b[i], b[maxIndex]);
        }
        for (int j = i + 1; j < size; j++)
        {
            const double factor = matrix[j][i] / matrix[i][i];
            for (int k = i; k < size; k++)
                matrix[j][k] -= factor * matrix[i][k];
            b[j] -= factor * b[i];
        }
    }
    std::vector<double> result(size, 0.0);
    for (int i = size - 1; i >= 0; i--)
    {
        double sum = 0;
        for (int j = i + 1; j < size; j++)
            sum += matrix[i][j] * result[j];
        result[i] = (b[i] - sum) / matrix[i][i];
    }
    std::cout << "\nТреугольный вид матрицы" << std::endl;
    printTwo(matrix);
    return result;
}

```

```cpp
bool convergence(std::vector<double>& x, std::vector<double>& newX, double eps)
{
    for (int i = 0; i < x.size(); i++)
    {
        if (abs(newX[i] - x[i]) > eps)
            return false;
    }
    return true;
}

std::vector<double> metodZeidela(std::vector<std::vector<double>> matrix, std::vector<double> b)
{
    const int size = matrix.size();
    const int maxItetation = 1000;
    const double eps = 0.00000000001;
    std::vector<double> penultimate_ittetation;
    std::vector<double> x(size, 0);
    for (int i = 0; i < maxItetation; i++)
    {
        std::vector<double> newX = x;
        for(int j = 0; j < size; j++)
        {
            double sum = 0;
            for (int k = 0; k < size; k++)
            {
                if (j != k)
                    sum += matrix[j][k] * newX[k];
            }
            newX[j] = (b[j] - sum) / matrix[j][j];
        }
        if (convergence(x, newX, eps))
        {
            std::cout << "\nМетод сошелся на " << (i + 1) << " итерации";
            std::cout << "\nПредпоследнее приближение" << std::endl;
            for (double k : penultimate_ittetation) {
                std::cout << k << std::endl;
            }
            return newX;
        }
        penultimate_ittetation = x;
        x = newX;
    }
    std::cout << "\nМетод не сошелся на " << maxItetation << " итераций";
    return x;
}

std::vector<double> metodSimpleIterations(std::vector<std::vector<double>> matrix, std::vector<double> b)
{
    const int size = matrix.size();
    const int maxItetation = 1000;
    const double eps = 0.00000000001;
    std::vector<double> x = { 0.0, 0.0, 0.0, 0.0 };
    for (int i = 0; i < maxItetation; i++)
    {
        std::vector<double> xOld = x;
        for (int j = 0; j < size; j++)
        {
            double s = 0;
            for (int k = 0; k < size; k++)
            {
                if (k != j)
                    s += matrix[j][k] * xOld[k];
            }
            x[j] = (b[j] - s) / matrix[j][j];
        }
        double norm = 0;
        for (int j = 0; j < size; j++)
            norm += abs(x[j] - xOld[j]);
        if (norm < eps)
        {
            std::cout << "\nМетод сошелся на " << (i + 1) << " итерации";
            return x;
        }
    }
    std::cout << "\nМетод не сошелся на " << maxItetation << " итераций";
    return std::vector<double> {};
}
```

```cpp
int main() {
    setlocale(LC_ALL, "Russian");
    std::cout.setf(std::ios::fixed);
    std::cout.precision(16);
    std::vector<std::vector<double>> matrix{
        {2.12, 0.42, 1.34, 0.88},
        {0.42, 3.95, 1.87, 0.43},
        {1.34, 1.87, 2.98, 0.46},
        {0.88, 0.43, 0.46, 4.44}
    };
    std::vector<double> b{ 11.172, 0.115, 9.009, 9.349 };
    std::vector<double> answer{ 3.7, -1.5, 2.1, 1.3 };

    std::cout << "Метод Гаусса" << std::endl;
    std::vector<double> gaussResult = metodGauss(matrix, b);

    std::cout << "\nРешение методом Гаусса" << std::endl;
    printOne(gaussResult); std::cout << std::endl;

    std::cout << "\nПогрешность метода Гаусса" << std::endl;
    calculationError(gaussResult, answer); std::cout << std::endl;

    std::cout << "Метод Зейделя" << std::endl;
    std::vector<double> zeidelResult = metodZeidela(matrix, b);

    std::cout << "\nРешение методом Зейделя" << std::endl;
    printOne(zeidelResult); std::cout << std::endl;

    std::cout << "\nПогрешность метода Зейделя" << std::endl;
    calculationError(zeidelResult, answer); std::cout << std::endl;

    std::cout << "Метод Простых итераций" << std::endl;
    std::vector<double> simpleIterationResult = metodSimpleIterations(matrix, b);

    std::cout << "\nРешение методом Простых итераций" << std::endl;
    printOne(simpleIterationResult); std::cout << std::endl;

    std::cout << "\nПогрешность метода Простых итераций" << std::endl;
    calculationError(simpleIterationResult, answer); std::cout << std::endl;
}
```