ОТЧЁТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

ЧИСЛЕННОЕ РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ
(Вариант 9)

Выполнил студент 3 курса ПМиИ
Кондратьев Виталий

***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*
9	$1{,}85 \quad 0{,}70 \quad -0{,}12 \quad -0{,}18$ $0{,}16 \quad 0{,}19 \quad 0{,}79 \quad 0{,}11$ $1{,}13 \quad 2{,}77 \quad 0{,}18 \quad -0{,}20$ $1{,}14 \quad 1{,}01 \quad 0{,}55 \quad 3{,}22$	$8{,}41$ $-0{,}23$ $13{,}91$ $9{,}58$	3 4 -2 1

Метод Гаусса с выбором главного элемента

Решение:

```
Точное решение:
3 4 -2 1
Martix A for method Gauss:
1.85    0.7     -0.12   -0.18
0       2.34243 0.253297        -0.0900541
0       0       0.786379        0.130545
0       0       0       3.25997
```

```
Метод Гаусса:
2.99999999999999956 4 -2 1.00000000000000022
Погрешность вычислений:
0.00000000000000444 0.00000000000000000 0.00000000000000000 0.00000000000000222
```

Метод Зейделя

Решение:

```
Преобразованная матрица A для метода Зейделя:
1.85000000000000089    0.00000000000000000    0.00000000000000000    0.00000000000000000
0.00000000000000000    2.34243243243243259    0.00000000000000000    0.00000000000000000
0.00000000000000000    0.00000000000000000    0.78637937002422986    0.00000000000000000
0.00000000000000000    0.00000000000000000    0.00000000000000000    3.25997275348284443
Преобразованная матрица b для метода Зейделя:
5.54999999999999822
9.36972972972973039
-1.57275874004845972
3.25997275348284532

 Число итераций для метода Зейделя: 82

3 последних итераций метода Зейделя:
3.00000000000000444    4.00000000000000888    -2.00000000000002220   1.00000000000000222
3.00000000000000444    4.00000000000000888    -2.00000000000002220   1.00000000000000222
3.00000000000000444    4.00000000000000888    -2.00000000000002220   1.00000000000000222

Метод Зейделя:
3.00000000000000444 4.00000000000000888 -2.00000000000002220 1.00000000000000222
Погрешность вычислений:
0.00000000000000444 0.00000000000000888 0.00000000000002220 0.00000000000000222
```

Вывод

Сравнение метода Гаусса с выбором главного элемента и метода Зейделя

Метод Гаусса с выбором главного элемента и метод Зейделя – два популярных метода решения систем линейных уравнений. Каждый из них обладает своими преимуществами и недостатками.

Точность:

• *Метод Гаусса с выбором главного элемента обеспечивает более высокую точность* решения, чем метод Зейделя. Это связано с тем, что метод Гаусса не использует деление на малые числа, что может привести к **ошибкам округления**.

Скорость:

• *Метод Зейделя*, как правило, работает *быстрее*, чем метод Гаусса, особенно для *хорошо обусловленных систем* с диагональным преобладанием.

Сходимость:

• *Метод Зейделя* может *не сходиться* для некоторых систем линейных уравнений.

• *Метод Гаусса с выбором главного элемента* всегда сходится, но может требовать *больше итераций*, чем метод Зейделя.

Выбор метода:

• *Метод Гаусса с выбором главного элемента* рекомендуется использовать, когда *требуется высокая точность* решения.

• *Метод Зейделя* может быть более подходящим, когда *важна скорость*, а требования к точности не так высоки.

Важно:

• Для достижения *оптимальной сходимости* метода Зейделя может потребоваться *настройка параметров*.

В целом:

• Выбор метода решения системы линейных уравнений зависит от *конкретных требований* к решению, таких как *точность, скорость и устойчивость* метода.

Дополнительные сведения:

• Для *более подробного сравнения* методов Гаусса и Зейделя рекомендуется ознакомиться с соответствующей литературой.

• *Выбор оптимального метода* для конкретной задачи может быть выполнен с помощью *экспериментального сравнения* различных методов.

ПРИЛОЖЕНИЕ

```
#include <iostream>
#include <vector>
#include <cmath>
#include <numeric>
#include <queue>

std::vector<std::vector<double>> transpose(std::vector<std::vector<double>>& oldMatrix);

void printArray(std::vector<std::vector<double>> const& matrix)
{
    for (auto const& str : matrix)
    {
        for (auto const& elem : str)
            std::cout << elem << "\t";
        std::cout << std::endl;
    }
}

void printArray(std::vector<double> const& vect)
{
    for (auto const& elem : vect)
        std::cout << elem << "\t";
    std::cout << '\n';
}

std::vector <double> GaussMethod(std::vector<std::vector<double>> const& A_old, std::vector<std::vector<double>>
const& b_old)
{
    std::vector<std::vector<double>> A = A_old;
    std::vector<std::vector<double>> b = b_old;
    int mSize = A.size();

    for (int col = 0; col < mSize; ++col)
    {
        int maxN = 0;
        for (int i = col; i < mSize; i++)
        {
            if (std::abs(A[i][col]) > std::abs(A[maxN][col]))
                maxN = i;
        }
        std::swap(A[col], A[maxN]);
        std::swap(b[col], b[maxN]);

        for (int i = col + 1; i < mSize; ++i)
        {
            double m = A[i][col] / A[col][col];
            b[i][0] -= b[col][0] * m;
            for (int j = 0; j < mSize; ++j)
            {
                A[i][j] -= A[col][j] * m;
            }
        }
    }

    std::cout << "Martix A for method Gauss: " << '\n';
    printArray(A);

    std::vector<double> x(mSize, 0.0);
    x[3] = b[3][0] / A[3][3];
    x[2] = (b[2][0] - A[2][3] * x[3]) / A[2][2];
```

```
    x[1] = (b[1][0] - A[1][3] * x[3] - A[1][2] * x[2]) / A[1][1];
    x[0] = (b[0][0] - A[0][3] * x[3] - A[0][2] * x[2] - A[0][1]*x[1]) / A[0][0];

    return x;
}

long double vectorNorm(std::vector<double> const& vect)
{
    long double res = 0.0;
    for (double element : vect)
    {
        res += element * element;
    }
    return std::sqrt(res);
}

std::vector <std::vector<double>> multipleMatrixes(std::vector<std::vector<double>> const& first,
std::vector<std::vector<double>> const& second)
{
    int R1 = first.size(), C1 = first[0].size(), R2 = second.size(), C2 = second[0].size();
    std::vector<std::vector<double>> result(first.size(), std::vector<double>(second[0].size(), 0));
    for (int i = 0; i < R1; i++)
    {
        for (int j = 0; j < C2; j++)
        {
            result[i][j] = 0;
            for (int k = 0; k < R2; k++)
            {
                result[i][j] += first[i][k] * second[k][j];
            }
        }
    }
    return result;
}

static int const numOfLoggedAnswers = 3;

void addXtoQueue(std::vector<double> const& x, std::queue<std::vector<double>>& xQueue)
{
    if (xQueue.size() < numOfLoggedAnswers)
    {
        xQueue.push(x);
    }
    else
    {
        xQueue.pop();
        xQueue.push(x);
    }
}

std::vector<std::vector<double>> transpose(std::vector<std::vector<double>>& oldMatrix)
{
    std::vector<std::vector<double>> newMatrix(oldMatrix[0].size(), std::vector<double>(oldMatrix.size(), 0));

    for (int i = 0; i < oldMatrix.size(); ++i)
    {
        for (int j = 0; j < oldMatrix[0].size(); ++j)
        {
            newMatrix[j][i] = oldMatrix[i][j];
        }
    }
    return newMatrix;
```

```
}


std::vector <double> SeidelMethod(std::vector<std::vector<double>> const& A_old, std::vector<std::vector<double>>
const& b_old)
{
    std::vector<std::vector<double>> A = A_old;
    std::vector<std::vector<double>> b = b_old;
    int mSize = A.size();

    for (int col = 0; col < mSize; ++col)
    {
        int maxN = 0;
        for (int i = col; i < mSize; i++)
        {
            if (std::abs(A[i][col]) > std::abs(A[maxN][col]))
                maxN = i;
        }
        std::swap(A[col], A[maxN]);
        std::swap(b[col], b[maxN]);

        for (int i = col + 1; i < mSize; ++i)
        {
            double m = A[i][col] / A[col][col];
            b[i][0] -= b[col][0] * m;
            for (int j = 0; j < mSize; ++j)
            {
                A[i][j] -= A[col][j] * m;
            }
        }
    }

    for (int i = mSize - 1; i >= 0; i--)
    {
        for (int j = 0; j < mSize; ++j)
        {
            double multiplier = A[j][i] / A[i][i];
            for (int k = i + 1; k < mSize; ++k)
            {
                A[j][k] -= A[i][k] * multiplier;
            }
            if (i != j)
            {
                b[j][0] -= b[i][0] * multiplier;
            }
        }
    }


    for (int i = 0; i < mSize; ++i)
    {
        double sum = 0;
        for (int j = 0; j < mSize; ++j)
        {
            if (i != j)
            {
                sum += std::abs(A[i][j]);
            }
        }
        if (abs(A[i][i]) < sum)
        {
```

```
        std::cout << "! ! ! УСЛОВИЕ СХОДИМОСТИ метода Зейделя НЕ ВЫПОЛНЯЕТСЯ ! ! !" << std::endl;
        return{};
      }

    }

    std::cout << "Преобразованная матрица A для метода Зейделя: " << '\n';
    printArray(A);

    std::cout << "Преобразованная матрица b для метода Зейделя: " << '\n';
    printArray(b);

    std::vector<std::vector<double>> A_o = A_old;

    A = multipleMatrixes(transpose(A_o), A_old);
    b = multipleMatrixes(transpose(A_o), b_old);

    std::queue<std::vector<double>> xQueue;

    std::vector<double> x(mSize, 0.0);
    int numOfIterations = 0;
    long double eps = std::pow(10, -15);
    while (true)
    {
      ++numOfIterations;
      std::vector<double> x_old = x;
      for (int i = 0; i < mSize; i++)
      {
        double sum = 0.0;
        for (int j = 0; j < mSize; j++)
        {
          if (j < i)
          {
            sum += A[i][j] * x[j];
          }
          else if (j > i)
          {
            sum += A[i][j] * x_old[j];
          }
        }
        x[i] = (b[i][0] - sum) / A[i][i];
        addXtoQueue(x, xQueue);
      }
      std::vector<double> currencyVect(x.size());
      for (int i = 0; i < x.size(); i++)
      {
        currencyVect[i] = x[i] - x_old[i];
      }
      if (vectorNorm(currencyVect) < eps)
        break;
    }
    std::cout << "\n Число итераций для метода Зейделя: " << numOfIterations << '\n';
    std::cout << '\n' << numOfLoggedAnswers << " последних итераций метода Зейделя: " << '\n';

    while (!xQueue.empty())
    {
      printArray(xQueue.front());
      xQueue.pop();
    }

    return x;
}
```

```cpp
void showResult(std::vector<double> const& exactX, std::vector<double> const& computedX)
{
    for (auto const& e : computedX)
    {
        std::cout.precision(18);
        std::cout << e << " ";
    }
    std::cout << "\nПогрешность вычислений: \n";
    for (int i = 0; i < exactX.size(); i++)
    {
        std::cout.precision(18);
        std::cout << std::fixed << std::abs(exactX[i] - computedX[i]) << " ";
    }
    std::cout << '\n';
}

int main()
{
    setlocale(LC_ALL, "Russian");

    std::vector<std::vector<double>> A{ {1.85, 0.70, -0.12, -0.18} ,
            {0.16, 0.19, 0.79, 0.11},
            {1.13, 2.77, 0.18, -0.20},
            {1.14, 1.01, 0.55, 3.22} };

    std::vector<std::vector<double>> b{ { 8.41} , { -0.23}, {13.91}, {9.58} };


    std::vector<double> x{ 3,4,-2,1 };

    std::cout << "Точное решение: " << '\n';
    for (auto e : x)
    {
        std::cout << e << " ";
    }
    std::cout << '\n';

    std::vector<double> result = GaussMethod(A, b);
    std::cout << "\nМетод Гаусса: " << '\n';
    showResult(x, result);

    result = SeidelMethod(A, b);
    std::cout << "\nМетод Зейделя: " << '\n';
    showResult(x, result);
    return 0;
}
```