Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей
Кафедра Информатики
Дисциплина «Методы численного анализа»

ОТЧЕТ
к лабораторной работе №1
на тему:
«РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ (СЛАУ) МЕТОДОМ ГАУССА И С ПОМОЩЬЮ ЕГО МОДИФИКАЦИЙ»
БГУИР 6-05-0612-02

Выполнил студент группы 453503
МИХАЛКО Александр Николаевич

(дата, подпись студента)

Проверил доцент каф. Информатики
АНИСИМОВ Владимир Яковлевич

(дата, подпись преподавателя)

Минск 2025

Оглавление

Вариант 7

Цели выполнения задания

- изучить метод Гаусса и его модификации, составить алгоритм метода и программу его реализации, получить численное решение заданной СЛАУ;
- составить алгоритм решения СЛАУ указанными методами, применимый для организации вычислений на ЭВМ;
- составить программу решения СЛАУ по разработанному алгоритму;
- выполнить тестовые примеры и проверить правильность работы программы.

Краткие теоритические сведения

СЛАУ обычно записывается в виде

$$\sum_{j=1}^{n} a_{ij} x_j = b_i; i \le 1 \le n \quad , \text{ или коротко} \quad \boldsymbol{Ax} = \boldsymbol{b} \, , \qquad (1.1)$$

где

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \quad ; \quad \boldsymbol{a} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} \quad ; \quad \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix}.$$

Здесь $\boldsymbol{A}$ и $\boldsymbol{b}$ заданы, требуется найти $\boldsymbol{x}$.

Методы решения СЛАУ делятся на прямые и итерационные. Прямые методы дают в принципе точное решение(если не учитывать ошибок округления) за конечное число арифметических операций. Они просты и наиболее универсальны. Для хорошо обусловленных систем небольшого порядка n ≤ 200 применяются практически только прямые методы. Наибольшее распространение среди прямых методов получили метод Гаусса и его модификации.

Метод Гаусса включает в себя прямой (приведение расширенной матрицы к ступенчатому виду, то есть получение нулей под главной диагональю) и обратный (получение нулей над главной диагональю расширенной матрицы) ходы. Прямой ход и называется методом Гаусса, обратный - методом Гаусса-Жордана, который отличается от первого только последовательностью исключения переменных.

Рассмотрим сначала простейший вариант метода Гаусса, называемый схемой единственного деления.

Прямой ход состоит из $n - 1$ шагов исключения.
1-й шаг. Целью этого шага является исключение неизвестного x_1 из уравнений с номерами $i = 2, 3, \dots, n$. Предположим, что коэффициент $a_{11} \neq 0$. Будем называть его *главным элементом* 1-*го шага*.

Найдем величины

$$q_{i1} = a_{i1}/a_{11} \quad (i = 2, 3, \dots, n),$$

называемые *множителями* 1-*го шага*. Вычтем последовательно из второго, третьего, …, n-го уравнений системы первое уравнение, умноженное

соответственно на q_{21}, q_{31}, ..., q_{n1}. Это позволит обратить в нуль коэффициенты при x_1 во всех уравнениях, кроме первого. В результате получим эквивалентную систему

$$\begin{aligned}
&a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n = b_1,\\
&a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + \ldots + a_{2n}^{(1)}x_n = b_2^{(1)},\\
&a_{32}^{(1)}x_2 + a_{33}^{(1)}x_3 + \ldots + a_{3n}^{(1)}x_n = b_3^{(1)},\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&a_{n2}^{(1)}x_2 + a_{n3}^{(1)}x_3 + \ldots + a_{nn}^{(1)}x_n = b_n^{(1)},
\end{aligned}$$

в которой $a_{ij}^{(1)}$ и $b_{ij}^{(1)}$ вычисляются по формулам

$$a_{ij}^{(1)} = a_{ij} - q_{i1}a_{1j} \quad , \quad b_i^{(1)} = b_i - q_{i1}b_1.$$

2-й шаг. Целью этого шага является ислючение неизвестного x_2 из уравнений с номерами $i = 3, 4, \ldots, n$. Пусть $a_{22}^{(1)} \neq 0$, где $a_{22}^{(1)}$ – коэффициент, называемый *главным* (или *ведущим*) *элементом 2-го шага*. Вычислим множители 2-го шага

$$q_{i2} = a_{i2}^{(1)} / a_{22}^{(1)} \quad (i = 3, 4, \ldots, n)$$

и вычтем последовательно из третьего, четвертого, …, n-го уравнений системы второе уравнение, умноженное соответственно на q_{32}, q_{42}, …, q_{n2}. В результате получим систему

$$\begin{aligned}
&a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n = b_1,\\
&a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + \ldots + a_{2n}^{(1)}x_n = b_2^{(1)},\\
&a_{33}^{(2)}x_3 + \ldots + a_{3n}^{(2)}x_n = b_3^{(2)},\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&a_{n3}^{(2)}x_3 + \ldots + a_{nn}^{(2)}x_n = b_n^{(2)}.
\end{aligned}$$

Здесь коэффициенты $a_{ij}^{(2)}$ и $b_j^{(2)}$ вычисляются по формулам

$$a_{ij}^{(2)} = a_{ij}^{(1)} - q_{i2}a_{2j}^{(1)} \quad , \quad b_i^{(2)} = b_i^{(1)} - q_{i2}b_2^{(1)}.$$

Аналогично проводятся остальные шаги. Опишем очередной k-й шаг.

k-й шаг. В предположении, что *главный* (*ведущий*) *элемент* k-го шага $a_{kk}^{(k-1)}$ отличен от нуля, вычислим *множители k-го шага*

$$q_{ik} = a_{ik}^{(k-1)} / a_{kk}^{(k-1)} \quad (i = k + 1, \ldots, n)$$

и вычтем последовательно из $(k + 1)$-го, …, n-го уравнений полученной на предыдущем шаге системы k-е уравнение, умноженное соответственно на $q_{k+1,k}$, $q_{k+2,k}$, …, q_{nk}.

После $(n - 1)$-го шага исключения получим систему уравнений

$$
\begin{array}{llllll}
a_{11}x_1 + & a_{12}x_2 & + a_{13}x_3 & + \ldots + & a_{1n}x_n = & b_1 \\
 & a_{22}^{(1)}x_2 + & a_{23}^{(1)}x_3 & + \ldots + & a_{2n}^{(1)}x_n = & b_2^{(1)} \\
 & & a_{33}^{(2)}x_3 & + \ldots + & a_{3n}^{(2)}x_n = & b_3^{(2)} \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
 & & & & a_{nn}^{(n-1)}x_n = & b_n^{(n-1)}
\end{array}
$$

матрица $A^{(n-1)}$ которой является верхней треугольной. На этом вычисления прямого хода заканчиваются.

Обратный ход. Из последнего уравнения системы находим x_n. Подставляя найденное значение x_n в предпоследнее уравнение, получим x_{n-1}. Осуществляя обратную подстановку, далее последовательно находим x_{n-2}, …, x_1. Вычисления неизвестных здесь проводятся по формулам

$$x_n = b_n^{(n-1)} / a_{nn}^{(n-1)},$$

$$x_k = (b_n^{(k-1)} - a_{k,k+1}^{(k-1)}x_{k+1} - \ldots - a_{kn}^{(k-1)}x_n) / a_{kk}^{(k-1)}, (k = n-1, \ldots, 1).$$

Необходимость отличия от 0 главных элементов. Заметим, что вычисление множителей, а также обратная подстановка требуют деления на главные элементы $a_{kk}^{(k-1)}$. Поэтому если один из главных элементов оказывается равным нулю, то схема не может быть реализована. Здравый смысл подсказывает, что и в ситуации, когда все главные элементы отличны от нуля, но среди них есть близкие к нулю, возможен неконтролируемый рост погрешности.

Метод Гаусса с выбором главного элемента по столбцу (схема частичного выбора). На k-м шаге прямого хода коэффициенты уравнений системы с номерами $i = k + 1, \ldots, n$ преобразуются по формулам
$a_{ij}^{(k)} = a_{ij}^{(k-1)} - q_{ik}a_{kj}$, $b_i^{(k)} = b_i^{(k-1)} - q_{ik}b_k^{(k-1)}$, $i = k + 1, \ldots, n$.

Интуитивно ясно, что во избежание сильного роста коэффициентов системы и связанных с этим ошибок нельзя допускать появления больших множителей q_{ik}.

В методе Гаусса с выбором главного элемента по столбцу гарантируется, что $|q_{ik}| \le 1$ для всех $k = 1, 2, \ldots, n-1$ и $i = k + 1, \ldots, n$. Отличие этого варианта метода Гаусса от схемы единственного деления заключается в том, что на k-м шаге исключения в качестве главного элемента выбирают максимальный по модулю коэффициент $a_{i_k k}$ при неизвестной x_k в уравнениях с номерами $i = k + 1, \ldots, n$. Затем соответствующее выбранному коэффициенту уравнение с номером i_k меняют местами с k-м уравнением системы для того, чтобы главный элемент занял место коэффициента $a_{kk}^{(k-1)}$. После этой перестановки исключение неизвестного x_k производят, как в схеме единственного деления.

Метод Гаусса с выбором главного элемента по всей матрице (схема полного выбора). В этой схеме допускается нарушение естественного порядка исключения неизвестных. На 1-м шаге метода среди элементов a_{ij} определяют максимальный по модулю элемент $a_{i_1 j_1}$. Первое уравнение

системы и уравнение с номером i_1 меняют местами. Далее стандартным образом производят исключение неизвестного x_{i_1} из всех уравнений, кроме первого.

На k-м шаге метода среди коэффициентов $a_{ij}^{(k-1)}$ при неизвестных в уравнениях системы с номерами $i = k, \ldots, n$ выбирают максимальный по модулю коэффициент $a_{i_k j_k}^{(k-1)}$. Затем k-е уравнение и уравнение, содержащее найденный коэффициент, меняют местами и исключают неизвестное x_{j_k} из уравнений с номерами $i = k + 1, \ldots, n$.

На этапе обратного хода неизвестные вычисляют в следующем порядке: $x_{j_n}, x_{j_{n-1}}, \ldots, x_{j_1}$.

Задание

Методом Гаусса и методом выбора главного элемента найти с точностью 0,0001 численное решение системы $Ax=b$,

где A = kC + D, A – исходная матрица для расчёта, k – номер варианта (0-15), матрицы C, D и вектор свободных членов b задаются ниже.

Вариант 7

Исходные данные:

Вектор **b**= $(4{,}2;\ 4{,}2;\ 4{,}2\ ;\ 4{,}2;\ 4{,}2)^{T}$,

$$C=\begin{bmatrix} 0{,}2 & 0 & 0{,}2 & 0 & 0 \\ 0 & 0{,}2 & 0 & 0{,}2 & 0 \\ 0{,}2 & 0 & 0{,}2 & 0 & 0{,}2 \\ 0 & 0{,}2 & 0 & 0{,}2 & 0 \\ 0 & 0 & 0{,}2 & 0 & 0{,}2 \end{bmatrix},\quad D=\begin{bmatrix} 2{,}33 & 0{,}81 & 0{,}67 & 0{,}92 & -0{,}53 \\ -0{,}53 & 2{,}33 & 0{,}81 & 0{,}67 & 0{,}92 \\ 0{,}92 & -0{,}53 & 2{,}33 & 0{,}81 & 0{,}67 \\ 0{,}67 & 0{,}92 & -0{,}53 & 2{,}33 & 0{,}81 \\ 0{,}81 & 0{,}67 & 0{,}92 & -0{,}53 & 2{,}33 \end{bmatrix}.$$

Полученные результаты будем сверять с решением, полученным используя модуль LinearAlgebra и команду LinearSolve:

```
X := LinearSolve(A, B);
```

Для исходных данных получим следующий ответ:

$X := LinearSolve(A, b);$

$$X:=\begin{bmatrix} 0.802566428324869 \\ 0.803244877358726 \\ 0.278772270892477 \\ 0.372662837682981 \\ 0.686999131562451 \end{bmatrix}$$

Программная реализация

Задача решена в общем виде для матрицы A размером $n \times m$, рассмотрены случаи, когда $n < m$, $n = m$, $n < m$, в первом методе Гаусса предусмотрен алгоритм, как избежать ситуации, когда в матрице A путем преобразований получается, что элемент $A[k][k] = 0.0$, где $0 \leq k \leq \min(n, m)$. В втором методе, дополнительно решена проблема, когда весь столбец начиная с элемента k, где $0 \leq k \leq \min(n, m)$, состоит из нулей.

Случай, когда $n < m$, ведет к тому, что система не доопределена и возможно бесконечно много решений, случай, $n = m$, полностью приспособлен к решению, одним из трех методов, вариант, когда $n < m$, решается при n = m, однако, как например в последнем методе поиск ведется по всей матрице, просто после нахождения решения также проверяются дополнительно n-m уравнений на соответствие решению в случае, не соответствия, приходим к выводу, что система не имеет решений.

Для первого метода механизм избегания деления на ноль, подразумевает проверку, что в конкретное уравнение содержит коэффициент отличный от нуля, после чего два столбца матрицы A меняются местами, и это изменение записывается в дополнительный массив, чтобы потом найти соответствующее решение, в противном случае идет проверка на существование решений.

Для второго метода была разработан механизм проверки на существование решения и возможность получения бесконечно много решений, в случае кода весь столбец начиная с элемента k, где $0 \leq k \leq \min(n, m)$, состоит из нулей, который начиная с k-той строки и k-столбца проверяет существует ли строка содержащая все коэффициенты со значением ноль, когда значение в матрице b в этой строке не нулевое. В случае нахождения такого случая можно сделать вывод, что система не имеет решений.

В реализации третьего метода, также рассмотрен случай работы с матрицей содержащей все нулевые элементы начиная A[k][k], проверяя k-ую строку в матрице b.

Три алгоритма для трех методов были реализованы в системе компьютерной алгебры Maple с использованием встроенного языка программирования, а также на высокоуровневом языке программирования

C++. Также на языке C++ была реализована базовый функционал для работы с матрицами, включая сложения матриц, умножения матриц и умножения матрицы на элемент, принадлежащий полю действительных чисел, а также структурированный вывод содержимого матриц.

1. Метод Гаусса

Матрица A, полученная в результате вычисления A=7*C*D в Maple:

$$\begin{bmatrix} 3.73000000000000 & 0.810000000000000 & 2.07000000000000 & 0.920000000000000 & -0.530000000000000 \\ -0.530000000000000 & 3.73000000000000 & 0.810000000000000 & 2.07000000000000 & 0.920000000000000 \\ 2.32000000000000 & -0.530000000000000 & 3.73000000000000 & 0.810000000000000 & 2.07000000000000 \\ 0.670000000000000 & 2.32000000000000 & -0.530000000000000 & 3.73000000000000 & 0.810000000000000 \\ 0.810000000000000 & 0.670000000000000 & 2.32000000000000 & -0.530000000000000 & 3.73000000000000 \end{bmatrix}$$

Код программы в Maple:

```
basicGauss := proc(A::Matrix, b::Vector)
  local n, m, colIndex, k, i, j, factor, sum, x, xCorrect, EPS, found, copy_A, copy_b, tmp;
  with(LinearAlgebra);

  EPS := 1e-10;
  n := RowDimension(A);
  m := ColumnDimension(A);

  copy_A := copy(A);
  copy_b := copy(b);

  if n < m then
    error "Система недоопределена: возможно бесконечно много решений";
  end if;

  colIndex := [seq(i, i = 1..m)];

  for k from 1 to m do
    if abs(A[k, k]) < EPS then
      found := false;
      for j from k + 1 to m do
        if abs(A[k, j]) > EPS then
          for i from 1 to n do
            tmp := A[i, k];
            A[i, k] := A[i, j];
            A[i, j] := tmp;
          end do;
          tmp := colIndex[k];
          colIndex[k] := colIndex[j];
          colIndex[j] := tmp;
          found := true;
          break;
        end if;
      end do;


          if not found then
            if abs(b[k]) > EPS then
              error "Система несовместна (нет решений)";
            else
              error "Система имеет бесконечно много решений";
            end if;
          end if;
        end if;

        for i from k + 1 to n do
          factor := A[i, k] / A[k, k];
          for j from k to m do
            A[i, j] := A[i, j] - factor * A[k, j];
          end do;
          b[i] := b[i] - factor * b[k];
        end do;
      end do;
```

```
  x := Vector(m, fill = 0.0);

  for i from m to 1 by -1 do
    sum := b[i];
    for j from i + 1 to m do
      sum := sum - A[i,j]*x[j];
    end do;

    if abs(A[i, i]) < EPS then
      if abs(sum) < EPS then
        error "Система имеет бесконечно много решений";
      else
        error "Система несовместна";
      end if;
    end if;
    x[i] := sum / A[i, i];
  end do;

  xCorrect := Vector(m, fill = 0.0);
  for i from 1 to m do
    xCorrect[colIndex[i]] := x[i];
  end do;
  x := xCorrect;

  for i from m + 1 to n do
    sum := 0;
    for j from 1 to m do
      sum := sum + copy_A[i,j]*x[j];
    end do;
    if abs(sum - copy_b[i]) > EPS then
      error "Система несовместна";
    end if;
  end do;

  return x;
end proc:
```

Матрица A, полученная в результате вычисления A=7*C*D на C++:

```
3.73 0.81 2.07 0.92 -0.53
-0.53 3.73 0.81 2.07 0.92
2.32 -0.53 3.73 0.81 2.07
0.67 2.32 -0.53 3.73 0.81
0.81 0.67 2.32 -0.53 3.73
```

Код программы на C++:

```cpp
bool basicGauss(std::vector<std::vector<double>>& A, std::vector<double>& b, std::vector<double>& x) {
    int n = A.size();
    int m = A[0].size();

    std::vector<std::vector<double>>& copy_A = A;
    std::vector<double>& copy_b = b;


    if (n < m) {
        std::cerr << "Система недоопределена: возможно бесконечно много решений\n";
        return false;
    }

    std::vector<int> colIndex(m);
    for (int i = 0; i < m; i++) colIndex[i] = i;

    for (int k = 0; k < m; k++) {
        if (fabs(A[k][k]) < EPS) {
            bool found = false;
            for (int j = k + 1; j < m; j++) {
                if (fabs(A[k][j]) > EPS) {
                    for (int i = 0; i < n; i++) std::swap(A[i][k], A[i][j]);
                    std::swap(colIndex[k], colIndex[j]);
                    found = true;
                    break;
                }
            }
            if (!found) {
                if (fabs(b[k]) > EPS) {
                    std::cerr << "Система несовместна (нет решений)\n";
                    return false;
                }
                else {
                    std::cerr << "Система имеет бесконечно много решений\n";
                    return false;
                }
            }
        }

        for (int i = k + 1; i < n; i++) {
            double factor = A[i][k] / A[k][k];
            for (int j = k; j < m; j++) {
                A[i][j] -= factor * A[k][j];
            }
            b[i] -= factor * b[k];
        }
    }
```

```cpp
    x.assign(m, 0.0);
    for (int i = m - 1; i >= 0; i--) {
        double sum = b[i];
        for (int j = i + 1; j < m; j++) {
            sum -= A[i][j] * x[j];
        }
        if (fabs(A[i][i]) < EPS) {
            if (fabs(sum) < EPS) {
                std::cerr << "Система имеет бесконечно много решений\n";
            }
            else {
                std::cerr << "Система несовместна\n";
            }
            return false;
        }
        x[i] = sum / A[i][i];
    }

    std::vector<double> xCorrect(m);
    for (int i = 0; i < m; i++) {
        xCorrect[colIndex[i]] = x[i];
    }
    x = xCorrect;


    for (int i = m; i < n; i++) {
        double sum = 0;
        for (int j = 0; j < m; j++) {
            sum += copy_A[i][j] * x[j];
        }
        if (fabs(sum - copy_b[i]) > EPS) {
            std::cerr << "Система несовместна\n";
            return false;
        }
    }

    return true;
}
```

Результат решения заданной системы вы можете увидеть в тестовом

примере 1.

2. Метод Гаусса с выбором главного элемента по столбцу

Код программы в Maple:

```
MaxElementInColumnGauss := proc(A::Matrix, b::Vector)
    local n, m, k, i, j, maxRow, factor, x, sum;
    n := LinearAlgebra:-RowDimension(A);
    m := LinearAlgebra:-ColumnDimension(A);
    if n < m then print("Система недоопределена: возможно бесконечно много решений"); return NULL end if;
    for k to m do
        maxRow := k;
        for i from k to n do if abs(A[maxRow, k]) < abs(A[i, k]) then maxRow := i end if end do;
        if abs(A[maxRow, k]) < 1.·10^-12 then
            for i from k to n do
                if add(abs(A[i,j]),j = k..m) < 1.·10^-12 then if abs(b[i]) < 1.·10^-12 then print("Система имеет бесконечно много решений") else print("Система несовместна") end if; return NULL end if
            end do
        end if;
        if maxRow <> k then A[[k, maxRow]] := A[[maxRow, k]]; b[[k, maxRow]] := b[[maxRow, k]] end if;
        for i from k + 1 to n do
            factor := A[i, k]/A[k, k]; for j from k to m do A[i,j] := A[i,j] - factor*A[k,j] end do; b[i] := b[i] - factor*b[k]
        end do
    end do;
    x := Vector(m, 0);
    for i from m by -1 to 1 do
        sum := b[i];
        for j from i + 1 to m do sum := sum - A[i,j]*x[j] end do;
        if abs(A[i, i]) < 1.·10^-12 then if abs(sum) < 1.·10^-12 then print("Система имеет бесконечно много решений") else print("Система несовместна") end if; return NULL end if;
        x[i] := sum/A[i, i]
    end do;
    for i from m + 1 to n do sum := add(A[i,j]*x[j],j = 1..m); if 1.·10^-12 < abs(sum - b[i]) then print("Система несовместна"); return NULL end if end do;
    return x
end proc
```

Код программы на C++:

```cpp
bool MaxElementInColumnGauss(std::vector<std::vector<double>>& A, std::vector<double>& b, std::vector<double>& x) {
    int n = A.size();
    int m = A[0].size();

    if (n < m) {
        std::cerr << "Система недоопределена: возможно бесконечно много решений\n";
        return false;
    }

    for (int k = 0; k < m; k++) {
        int maxRow = k;
        for (int i = k; i < n; i++) {
            if (fabs(A[i][k]) > fabs(A[maxRow][k])) {
                maxRow = i;
            }
        }

        if (fabs(A[maxRow][k]) < EPS) {
            bool isZero = true;
            for (int l = k; l < n; l++) {
                for (int v = k; v < n; v++) {
                    if (A[l][v] != 0) isZero = false;
                }
                if (isZero && fabs(b[l]) < EPS) {
                    std::cerr << "Система имеет бесконечно много решений\n";
                    return false;
                }
            }

            std::cerr << "Система имеет бесконечно много решений\n";
            return false;
        }

        if (maxRow != k) {
            std::swap(A[k], A[maxRow]);
            std::swap(b[k], b[maxRow]);
        }

        for (int i = k + 1; i < n; i++) {
            double factor = A[i][k] / A[k][k];
            for (int j = k; j < m; j++) {
                A[i][j] -= factor * A[k][j];
            }
            b[i] -= factor * b[k];
        }
    }
```

```
    x.assign(m, 0.0);
    for (int i = m - 1; i >= 0; i--) {
        double sum = b[i];
        for (int j = i + 1; j < m; j++) {
            sum -= A[i][j] * x[j];
        }
        if (fabs(A[i][i]) < EPS) {
            if (fabs(sum) < EPS) {
                std::cerr << "Система имеет бесконечно много решений\n";
            }
            else {
                std::cerr << "Система несовместна\n";
            }
            return false;
        }
        x[i] = sum / A[i][i];
    }

    for (int i = m; i < n; i++) {
        double sum = 0;
        for (int j = 0; j < m; j++) {
            sum += A[i][j] * x[j];
        }
        if (fabs(sum - b[i]) > EPS) {
            std::cerr << "Система несовместна\n";
            return false;
        }
    }

    return true;
}
```

Результат решения заданной системы вы можете увидеть в тестовом примере 1.

3. Метод Гаусса с выбором главного элемента по всей матрице

Код программы в Maple:

```
MaxElementInMatrixGauss := proc(A :: Matrix, b :: Vector)
  local n, m, copy_A, copy_b, colIndex, x, xCorrect, EPS;
  local k, i, j, r, max, indexRow, indexColm, q, sum, tmp;

  EPS := 1e-9;
  n := RowDimension(A);
  m := ColumnDimension(A);

  copy_A := copy(A);
  copy_b := copy(b);

  if n < m then
    error "Система недоопределена: возможно бесконечно много решений";
  end if;

  colIndex := [seq(i, i = 1 ..m)];

  for k from 1 to m do
    max := A[k, k];
    indexRow := k;
    indexColm := k;

    for i from k to n do
      for j from k to m do
        if abs(max) < abs(A[i, j]) then
          max := A[i, j];
          indexRow := i;
          indexColm := j;
        end if;
      end do;
    end do;

    if abs(A[indexRow, indexColm]) < EPS then
      if abs(b[indexRow]) > EPS then
        error "Система несовместна (нет решений)";
      else
        error "Система имеет бесконечно много решений";
      end if;
    end if;

    if indexRow ≠ k then
      RowOperation(A, [k, indexRow], inplace = true);
      tmp := b[k];
      b[k] := b[indexRow];
      b[indexRow] := tmp;
    end if;

    if indexColm ≠ k then
      for r from 1 to n do
        tmp := A[r, k];
        A[r, k] := A[r, indexColm];
        A[r, indexColm] := tmp;
```

```
        end do;
        tmp := colIndex[k];
        colIndex[k] := colIndex[indexColm];
        colIndex[indexColm] := tmp;
      end if;

      for i from k + 1 to n do
        q := A[i, k] / A[k, k];
        for j from k to m do
          A[i,j] := A[i,j] - q*A[k,j];
        end do;
        b[i] := b[i] - q* b[k];
      end do;
    end do;

    x := Vector(m, fill = 0.0);
    for i from m to 1 by -1 do
      sum := b[i];
      for j from i + 1 to m do
        sum := sum - A[i,j]*x[j];
      end do;
      if abs(A[i, i]) < EPS then
        if abs(sum) < EPS then
          error "Система имеет бесконечно много решений";
        else
          error "Система несовместна";
        end if;
      end if;
      x[i] := sum / A[i, i];
    end do;

    xCorrect := Vector(m, fill = 0.0);
    for i from 1 to m do
      xCorrect[colIndex[i]] := x[i];
    end do;
    x := xCorrect;

    if n > m then
      for i from m + 1 to n do
        sum := 0;
        for j from 1 to m do
          sum := sum + copy_A[i,j]*x[j];
        end do;
        if abs(sum - copy_b[i]) > EPS then
          error "Система несовместна";
        end if;
      end do;
    end if;

    return x;
  end proc:
```

Код программы на C++:

```cpp
bool MaxElementInMatrixGauss(std::vector<std::vector<double>>& A, std::vector<double>& b, std::vector<double>& x) {
    int n = A.size();
    int m = A[0].size();

    std::vector<std::vector<double>>& copy_A = A;
    std::vector<double>& copy_b = b;

    if (n < m) {
        std::cerr << "Система недоопределена: возможно бесконечно много решений\n";
        return false;
    }


    std::vector<int> colIndex(m);
    for (int i = 0; i < m; i++) colIndex[i] = i;

    for (int k = 0; k < m; k++) {

        double max = A[k][k];
        int indexColm = k;
        int indexRow = k;

        for (int i = k; i < n; i++) {
            for (int j = k; j < m; j++) {
                if (fabs(max) < fabs(A[i][j])) {
                    max = A[i][j];
                    indexRow = i;
                    indexColm = j;
                }
            }
        }

        if (fabs(A[indexRow][indexColm]) < EPS) {
            if (fabs(b[indexRow]) > EPS) {
                std::cerr << "Система несовместна (нет решений)\n";
            }
            else {
                std::cerr << "Система имеет бесконечно много решений\n";
            }
            return false;
        }

        if (indexRow != k) {
            std::swap(A[k], A[indexRow]);
            std::swap(b[k], b[indexRow]);
        }
```

```
        if (indexColm != k) {
            for (int r = 0; r < m; r++) std::swap(A[r][k], A[r][indexColm]);
            std::swap(colIndex[k], colIndex[indexColm]);
        }

        for (int i = k + 1; i < n; i++) {
            double q = A[i][k] / A[k][k];
            for (int j = k; j < m; j++) {
                A[i][j] -= q * A[k][j];
            }

            b[i] -= q * b[k];
        }
    }

    x.assign(m, 0.0);
    for (int i = m - 1; i >= 0; i--) {
        double sum = b[i];
        for (int j = i + 1; j < m; j++) {
            sum -= A[i][j] * x[j];
        }
        if (fabs(A[i][i]) < EPS) {
            if (fabs(sum) < EPS) {
                std::cerr << "Система имеет бесконечно много решений\n";
            }
            else {
                std::cerr << "Система несовместна\n";
            }
            return false;
        }
        x[i] = sum / A[i][i];
    }

    std::vector<double> xCorrect(m);
    for (int i = 0; i < m; i++) {
        xCorrect[colIndex[i]] = x[i];
    }
    x = xCorrect;

    if (n > m) {
        for (int i = m; i < n; i++) {
            double sum = 0;
            for (int j = 0; j < m; j++) {
                sum += copy_A[i][j] * x[j];
            }
            if (fabs(sum - copy_b[i]) > EPS) {
                std::cerr << "Система несовместна\n";
                return false;
            }
        }
    }
    return true;
}
```

Результат решения заданной системы вы можете увидеть в тестовом примере 1.

Полученные результаты

Тестовый пример 1.

Матрица A, полученная в результате вычисления A=3C*D в Maple:

$$\begin{bmatrix} 3.73000000000000 & 0.810000000000000 & 2.07000000000000 & 0.920000000000000 & -0.530000000000000 \\ -0.530000000000000 & 3.73000000000000 & 0.810000000000000 & 2.07000000000000 & 0.920000000000000 \\ 2.32000000000000 & -0.530000000000000 & 3.73000000000000 & 0.810000000000000 & 2.07000000000000 \\ 0.670000000000000 & 2.32000000000000 & -0.530000000000000 & 3.73000000000000 & 0.810000000000000 \\ 0.810000000000000 & 0.670000000000000 & 2.32000000000000 & -0.530000000000000 & 3.73000000000000 \end{bmatrix}$$

Метод Гаусса	Метод Гаусса с выбором главного элемента по столбцу	Метод Гаусса с выбором главного элемента по всей матрице	Вычисление при помощи встроенной функции
0.8026	0.8026	0.8026	0.8026
0.8032	0.8032	0.8032	0.8032
0.2788	0.2788	0.2788	0.2788
0.3727	0.3727	0.3727	0.3727
0.687	0.687	0.687	0.687

.802566428324869	.802566428324869	.802566428324869	. 802566428324869
.803244877358725	.803244877358725	.803244877358726	.803244877358726
. .278772270892477	.278772270892477	.278772270892477	.278772270892477
.372662837682981	.372662837682981	.372662837682982	.372662837682981
.686999131562451	.686999131562451	.686999131562452	.686999131562451

Матрица A, полученная в результате вычисления A=3C*D на C++:

```
3.73 0.81 2.07 0.92 -0.53
-0.53 3.73 0.81 2.07 0.92
2.32 -0.53 3.73 0.81 2.07
0.67 2.32 -0.53 3.73 0.81
0.81 0.67 2.32 -0.53 3.73
```

Метод Гаусса	Метод Гаусса с выбором главного элемента по столбцу	Метод Гаусса с выбором главного элемента по всей матрице
0.8026	0.8026	0.8026
0.8032	0.8032	0.8032
0.2788	0.2788	0.2788
0.3727	0.3727	0.3727
0.687	0.687	0.687

0.802566	. 0.802566	0.802566
0.803245	0.803245	0.803245
0. 278772	0. 278772	0. 278772
0. 372663	0. 372663	0. 372663
0.686999	0.686999	0.686999

Тестовый пример 2.

Проверка на поведение методов при попытке решить систему, у которой нет решений

Исходная матрица, матрицы b имеет исходный вид:

$$M := \begin{bmatrix} 0. & 0. & 0. & 0. & 0. \\ -0.53 & 2.33 & 0.81 & 0.67 & 0.92 \\ 0.92 & -0.53 & 2.33 & 0.81 & 0.67 \\ 0.67 & 0.92 & -0.53 & 2.33 & 0.81 \\ 0.81 & 0.67 & 0.92 & -0.53 & 2.33 \end{bmatrix}$$

Все три метода сделали вывод, что система не имеет решений.

Тестовый пример 3.

Исходная матрица содержит m столбцов и n строк, причем m > n

$$M := \begin{bmatrix} 0. & 0. & 0. & 0. & 0. \\ -0.53 & 2.33 & 0.81 & 0.67 & 0.92 \\ 0.92 & -0.53 & 2.33 & 0.81 & 0.67 \\ 0.67 & 0.92 & -0.53 & 2.33 & 0.81 \end{bmatrix}$$

Все три метода сделали вывод, что система может иметь бесконечно много решений.

Реализация базовых операций над матрицами на C++:

Сложение двух матриц с проверкой условия на возможность операции:

```cpp
std::vector<std::vector<double>> sumMyMatrices(
    const std::vector<std::vector<double>>& A,
    const std::vector<std::vector<double>>& B) {

    int rowsA = A.size();
    int colsA = A[0].size();
    int rowsB = B.size();
    int colsB = B[0].size();

    if (rowsA != rowsB || colsA != colsB) {
        throw std::invalid_argument("Несовместимые размеры матриц для сложения");
    }

    std::vector<std::vector<double>> result(rowsA, std::vector<double>(colsB, 0.0));

    for (int i = 0; i < rowsA; i++) {
        for (int j = 0; j < colsA; j++) {
            result[i][j] = A[i][j] + B[i][j];
        }
    }

    return result;
}
```

Умножения элемента поля действительных чисел на матрицу:

```cpp
std::vector<std::vector<double>> multiplintKAndMatrix(
    const std::vector<std::vector<double>>& A, const double k)
{
    int rowsA = A.size();
    int colsA = A[0].size();

    std::vector<std::vector<double>> result(rowsA, std::vector<double>(colsA, 0.0));

    for (int i = 0; i < rowsA; i++) {
        for (int j = 0; j < colsA; j++) {
            result[i][j] = A[i][j] * k;
        }
    }

    return result;
}
```

Умножения двух матриц с проверкой на возможность операции:

```cpp
std::vector<std::vector<double>> ADDIRIONALmultiplayMatrics(const std::vector<std::vector<double>>& A,
    const std::vector<std::vector<double>>& B)
    {
        int rowsA = A.size();
        int colsA = A[0].size();
        int rowsB = B.size();
        int colsB = B[0].size();

        if (colsA != rowsB){
            throw std::invalid_argument("Несовместимые размеры матриц для умножения");
        }

        std::vector<std::vector<double>> result(rowsA, std::vector<double>(colsB, 0.0));

        for (int k = 0; k < rowsA; k++) {
            for (int j = 0; j < colsB; j++) {
                for (int i = 0; i < colsA; i++) {
                    result[k][j] += A[k][i] * B[i][j];
                }
            }
        }

        return result;

}
```

Дополнительные расчеты:

Норма матрицы — это способ измерить "размер" матрицы как оператора. Она показывает, насколько сильно матрица может растянуть вектор.

На практике используют стандартные нормы:

- **1-норма (столбцовая):**

$$\|A\|_1 = \max_j \sum_i |a_{ij}|$$

(максимальная сумма по столбцам).

- **∞-норма (строковая):**

$$\|A\|_\infty = \max_i \sum_j |a_{ij}|$$

(максимальная сумма по строкам).

Значения норм матрицы A:

```
ConditionNumber(A, 1);
                                                    8.808056532

> Norm(A, infinity);
                                                    9.46000000000000085
```

Число обусловленности в матрице A при A*x=b, определяется как

$$cond(A) = \left||A|\right| * \left||A^{-1}|\right|$$

Если $cond(A)$ близко к 1 значит система **хорошо обусловлена**, ошибки в данных почти не искажают решение. Если $cond(A)$ большое (например, 10^6) значит система **плохо обусловлена**, малые ошибки в исходных данных (округление, неточности) могут сильно исказить решение. То есть $cond(A)$**показывает чувствительность решения к погрешностям**.

Значения чисел обусловленности для матрицы A:

```
ConditionNumber(A, 1);    # в 1-норме
```

8.808056532

```
ConditionNumber(A, infinity); # в бесконечной норме
```

8.808056532

Невязка — это вектор:

$$r = A\tilde{x} - b$$

Он показывает, насколько найденное решение $\tilde{x}$ действительно удовлетворяет системе.

- Если $r = 0$, значит $\tilde{x}$ — точное решение.
- Если $r \neq 0$, значит решение приближённое.

Обычно смотрят **норму невязки**:

$$\|r\|, \quad \frac{\|r\|}{\|b\|} \quad \text{(относительная невязка)}$$

Невязка для решения путем применения первой схемы:

```
r := A • x − b;
```

$$r := \begin{bmatrix} 0. \\ -1.77635683940025\,10^{-15} \\ 0. \\ -8.88178419700125\,10^{-16} \\ 0. \end{bmatrix}$$

Относительная невязки:

```
Norm(r, 1) / Norm(b, 1);
```

$1.26882631\,4\,10^{-16}$

Посчитаем абсолютную ошибку отняв матрицу решения полученную нами и матрицу полученную встроенными методами и посчитав норму и посчитаем относительную ошибку разделив абсолютную ошибку на значение нормы матрицы ответа, полученного встроенными методами:

$$X := LinearSolve(A, b);$$

$$X := \begin{bmatrix} 0.802566428324869 \\ 0.803244877358726 \\ 0.278772270892477 \\ 0.372662837682981 \\ 0.686999131562451 \end{bmatrix}$$

$$x$$

$$\begin{bmatrix} 0.802566428324869 \\ 0.803244877358725 \\ 0.278772270892477 \\ 0.372662837682981 \\ 0.686999131562451 \end{bmatrix}$$

$$oureror := x - X$$

$$oureror := \begin{bmatrix} 1.11022302462516\,10^{-16} \\ -1.11022302462516\,10^{-16} \\ -5.55111512312578\,10^{-17} \\ 0. \\ 0. \end{bmatrix}$$

$$Norm(oureror, 1);$$

$$2.77555756156289135\,10^{-16}$$

$$\frac{Norm(oureror, 1)}{Norm(X, 1)}$$

$$9.42705871\,10^{-17}$$

Выводы

Таким образом в ходе выполнения работы были реализованы три метода Гаусса для решения системы линейных алгебраических уравнений, а именно схема единственного деления, схема частичного выбора и схема полного выбора. Задача была решения в общем виде для матрицы A размером m*n, был предложен метод, как избежать деления на ноль при использовании первой схемы, рассмотрены варианты, когда система не совместна или имеет бесконечно много решений, а также каждая схема была реализована в системе Maple с использованием встроенного языка программирования, а также на языке C++, в том числе на C++ были написаны дополнительно базовые операции над матрицами, которые уже

реализованы в Maple. В тестовом примере 1 были предоставлены результаты решения заданной системы тремя методами в Maple, на C++ и встроенным решением от Maple, при задании точности до 4 знаков после запятой, результаты во всех случаях совпадают. Также были рассчитаны норма матрицы, число обусловленности, невязку, абсолютную и относительную ошибки.