Федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
(ФПИиКТ)

Лабораторная работа №1 "Решение системы линейных алгебраических уравнений СЛАУ"

по дисциплине «Вычислительная математика»

Вариант 8

Выполнил(а): студент группы P3215
Зыков Иван Евгеньевич

Проверил(а): Малышева Татьяна Алексеевна

Санкт-Петербург
2025

Содержание

1 Цель

Найти решение системы линейных уравнений (СЛУ) методом Гаусса с выбором главного элемента по столбцам. Вычислить погрешность, сравнить результаты с готовыми решениями из встроенных библиотек.

2 Описание метода. Расчетные формулы

2.1 Метод Гаусса с выбором главного элемента по столбцам

Метод Гаусса заключается в приведении системы линейных уравнений ступенчатому виду с помощью элементарных преобразований. Выбор главного элемента по столбцам позволяет уменьшить ошибки округления.

2.2 Основные шаги метода

1. **Прямой ход**:
   - На каждом шаге выбирается максимальный по модулю элемент в текущем столбце.
   - Строка с главным элементом меняется местами с текущей строкой.
   - Выполняется исключение переменной из всех строк ниже текущей.
2. **Обратный ход**:
   - Начиная с последней строки, вычисляются значения неизвестных.

2.3 Расчетные формулы

1. **Выбор главного элемента**:

$$\text{max\_val} = \max_{i=k}^{n-1} |A[i,k]|$$

   где k — текущий шаг, n — размерность матрицы.

2. **Исключение переменной**:

$$\text{factor} = \frac{A[i,k]}{A[k,k]}$$

$$A[i,j] = A[i,j] - \text{factor} \cdot A[k,j]$$

$$B[i] = B[i] - \text{factor} \cdot B[k]$$

3. **Обратный ход**:

$$X[i] = \frac{B[i] - \sum_{j=i+1}^{n-1} A[i,j] \cdot X[j]}{A[i,i]}$$

3 Листинг программы

```
#!/bin/bash

error_message() {
    echo -e "\e[1;31m  /\_/\  \e[0m"
    echo -e "\e[1;31m ( T.T ) \e[0m"
    echo -e "\e[1;31m  > ^ <  \e[0m"
    echo -e "\e[1;31mОшибка: $1\e[0m" >&2
    exit 1
}

input_matrix() {
    read -p "Введите размерность матрицы (n): " n
    if ! [[ "$n" =~ ^[0-9]+$ ]] || [ "$n" -lt 1 ] || [ "$n" -gt 20 ]; then
        error_message "Размерность матрицы должна быть целым числом от 1 до 20."
    fi

    echo "Введите матрицу A построчно: "
    for ((i = 0; i < n; i++)); do
        read -p "Строка $((i + 1)): " row
        j=0
        for num in $row; do
            if ! [[ "$num" =~ ^-?[0-9]+(\.[0-9]+)?$ ]]; then
                error_message "Элемент '$num' не является числом."
            fi
            A[$i,$j]=$num
            ((j++))
        done
        if [ $j -ne "$n" ]; then
            error_message "Количество элементов в строке $((i + 1))
            должно быть равно $n."
        fi
    done

    echo -e "${A[0,0]}, ${A[0,1]} \n${A[1,0]}, ${A[1,1]}"

    echo "Введите вектор B: "
    read -p "Вектор B: " string
    j=0
    for num in $string; do
        if ! [[ "$num" =~ ^-?[0-9]+(\.[0-9]+)?$ ]]; then
            error_message "Элемент '$num' не является числом."
        fi
        B[$j]=$num
        ((j++))
    done
    if [ $j -ne "$n" ]; then
        error_message "Количество элементов в векторе B должно быть равно $n."
    fi
```

```
}

input_matrix_from_file() {
    read -p "Введите имя файла: " filename
    if [ ! -f "$filename" ]; then
        error_message "Файл '$filename' не найден."
    fi

    n=$(head -n 1 "$filename")
    if ! [[ "$n" =~ ^[0-9]+$ ]] || [ "$n" -lt 1 ] || [ "$n" -gt 20 ]; then
        error_message "Размерность матрицы в файле должна быть целым числом
        от 1 до 20."
    fi

    for ((i = 0; i < n; i++)); do
        row=$(awk -v i=$i 'NR==i+2 {print}' "$filename")
        j=0
        for num in $row; do
            if ! [[ "$num" =~ ^-?[0-9]+(\.[0-9]+)?$ ]]; then
                error_message "Элемент '$num' в файле не является числом."
            fi
            A[$i,$j]=$num
            ((j++))
        done
        if [ $j -ne "$n" ]; then
            error_message "Количество элементов в строке $((i + 1)) файла
            не равно $n."
        fi
    done

    B=($(awk -v n=$n 'NR==n+2 {print}' "$filename"))
    if [ ${#B[@]} -ne "$n" ]; then
        error_message "Количество элементов вектора B в файле не равно $n."
    fi
}

print_matrix() {
    for ((i = 0; i < n; i++)); do
        for ((j = 0; j < n; j++)); do
            printf "%8.4f " ${A[$i,$j]}
        done
        printf "| %8.4f\n" ${B[$i]}
    done
}

gauss_elimination() {
    for ((k = 0; k < n - 1; k++)); do
        max_row=$k
        max_val=${A[$k,$k]}
        for ((i = k + 1; i < n; i++)); do
```

```
            if (( $(echo "${A[$i,$k]} > $max_val" | bc -l) )); then
                max_row=$i
                max_val=${A[$i,$k]}
            fi
        done
        if (( max_row != k )); then
            for ((j = k; j < n; j++)); do
                temp=${A[$k,$j]}
                A[$k,$j]=${A[$max_row,$j]}
                A[$max_row,$j]=$temp
            done
            temp=${B[$k]}
            B[$k]=${B[$max_row]}
            B[$max_row]=$temp
        fi
        if (( $(echo "${A[$k,$k]} == 0" | bc -l) )); then
            error_message "Матрица вырождена. Решение невозможно."
        fi
        for ((i = k + 1; i < n; i++)); do
            factor=$(echo "scale=10; ${A[$i,$k]} / ${A[$k,$k]}" | bc -l)
            for ((j = k; j < n; j++)); do
                A[$i,$j]=$(echo "scale=10; ${A[$i,$j]} - $factor * ${A[$k,$j]}"
                | bc -l)
            done
            B[$i]=$(echo "scale=10; ${B[$i]} - $factor * ${B[$k]}" | bc -l)
        done
    done
}

calculate_determinant() {
    determinant=1
    for ((i = 0; i < n; i++)); do
        determinant=$(echo "scale=10; $determinant * ${A[$i,$i]}" | bc -l)
    done
    echo "Определитель матрицы: $determinant"
}

back_substitution() {
    for ((i = n - 1; i >= 0; i--)); do
        X[$i]=${B[$i]}
        for ((j = i + 1; j < n; j++)); do
            X[$i]=$(echo "scale=10; ${X[$i]} - ${A[$i,$j]} * ${X[$j]}" | bc -l)
        done
        X[$i]=$(echo "scale=10; ${X[$i]} / ${A[$i,$i]}" | bc -l)
    done
    echo "Вектор неизвестных:"
    for ((i = 0; i < n; i++)); do
        printf "X[%d] = %8.4f\n" $i ${X[$i]}
    done
}
```

```
calculate_residuals() {
    echo "Вектор невязок:"
    for ((i = 0; i < n; i++)); do
        residual=${B[$i]}
        for ((j = 0; j < n; j++)); do
            residual=$(echo "scale=10; $residual - ${A[$i,$j]} * ${X[$j]}" | bc -l)
        done
        printf "R[%d] = %8.4f\n" $i $residual
    done
}

calculate_rank() {
    local -n matrix=$1
    local rank=0
    for ((k = 0; k < n; k++)); do
        pivot_row=-1
        for ((i = k; i < n; i++)); do
            if (( $(echo "${matrix[$i,$k]} != 0" | bc -l) )); then
                pivot_row=$i
                break
            fi
        done
        if [ $pivot_row -ne -1 ]; then
            ((rank++))
            if [ $pivot_row -ne $k ]; then
                for ((j = 0; j < n; j++)); do
                    temp=${matrix[$k,$j]}
                    matrix[$k,$j]=${matrix[$pivot_row,$j]}
                    matrix[$pivot_row,$j]=$temp
                done
            fi
            for ((i = k + 1; i < n; i++)); do
                factor=$(echo "scale=10; ${matrix[$i,$k]} / ${matrix[$k,$k]}" |
                bc -l)
                for ((j = k; j < n; j++)); do
                    matrix[$i,$j]=$(echo "scale=10; ${matrix[$i,$j]} -
                    $factor * ${matrix[$k,$j]}" | bc -l)
                done
            done
        fi
    done
    echo $rank
}

check_system() {
    declare -A A_copy
    declare -a B_copy
    for ((i = 0; i < n; i++)); do
        for ((j = 0; j < n; j++)); do
```

```
            A_copy[$i,$j]=${A[$i,$j]}
        done
        B_copy[$i]=${B[$i]}
    done

    rank_A=$(calculate_rank A_copy)

    declare -A augmented_matrix
    for ((i = 0; i < n; i++)); do
        for ((j = 0; j < n; j++)); do
            augmented_matrix[$i,$j]=${A[$i,$j]}
        done
        augmented_matrix[$i,$n]=${B[$i]}
    done

    rank_augmented=$(calculate_rank augmented_matrix)

    if (( $(echo "$rank_A == $rank_augmented" | bc -l) )); then
        if (( $(echo "$rank_A == $n" | bc -l) )); then
            echo -e "\e[1;32m  /\_/\  \e[0m"
            echo -e "\e[1;32m ( ^.^ ) \e[0m"
            echo -e "\e[1;32m  > ^ <  \e[0m"
            echo -e "\e[1;32m Система имеет единственное решение! \e[0m"
        else
            error_message "Система имеет бесконечно много решений."
        fi
    else
        error_message "Система не имеет решений."
    fi
}

check_with_numpy() {
    echo "Проверка решения через numpy:"
    python3 - <<EOF
import numpy as np

A = np.array([$(for ((i = 0; i < n; i++)); do echo -n "["; for ((j = 0; j < n; j++));
do echo -n "${A[$i,$j]},"; done; echo -n "],"; done)])
B = np.array([$(for ((i = 0; i < n; i++)); do echo -n "${B[$i]},"; done)])

try:
    X_numpy = np.linalg.solve(A, B)
    det_numpy = np.linalg.det(A)
    print("Решение (numpy):", X_numpy)
    print("Определитель (numpy):", det_numpy)
except np.linalg.LinAlgError:
    print("Система не имеет решения или имеет бесконечно много решений.")
EOF
}
```

```
# Точка входа
echo -e "\e[1;35m  /\_/\  \e[0m"
echo -e "\e[1;35m ( o.o ) \e[0m"
echo -e "\e[1;35m  > ^ <  \e[0m"
echo "Выберите способ ввода матрицы:"
echo "1 - С клавиатуры"
echo "2 - Из файла"
read choice

declare -A A
declare -a B

case $choice in
    1) input_matrix ;;
    2) input_matrix_from_file ;;
    *) error_message "Неверный выбор." ;;
esac

echo "Исходная матрица:"
print_matrix

check_system

if (( $(echo "$rank_A == $rank_augmented" | bc -l) )); then
    gauss_elimination
    echo "Треугольная матрица:"
    print_matrix
    calculate_determinant
    back_substitution
    calculate_residuals
fi

check_with_numpy

echo -e "\e[1;32m  /\_/\  \e[0m"
echo -e "\e[1;32m ( ^.^ ) \e[0m"
echo -e "\e[1;32m  > ^ <  \e[0m"
echo -e "\e[1;32m Ура! Мы справились! \e[0m"
```

4 Примеры работы скрипта

4.1 Пример 1

4.1.1 Входные данные

Матрица A:

$$\begin{pmatrix} 2 & 1 \\ 1 & 3 \end{pmatrix}$$

Вектор B:

$$\begin{pmatrix} 5 \\ 7 \end{pmatrix}$$

4.1.2 Результат

```
  /\_/\
 ( ^.^ )
  > ^ <
 Система имеет единственное решение!
Треугольная матрица:
  2.0000   1.0000 |   5.0000
  0.0000   2.5000 |   4.5000
Определитель матрицы: 5.0000000000
Вектор неизвестных:
X[0] =   1.6000
X[1] =   1.8000
Вектор невязок:
R[0] =   0.0000
R[1] =   0.0000
Проверка решения через numpy:
Решение (numpy): [1.6 1.8]
Определитель (numpy): 5.000000000000001
  /\_/\
 ( ^.^ )
  > ^ <
 Ура! Мы справились!
```

4.2 Пример 2

4.2.1 Входные данные

Матрица A:

$$\begin{pmatrix} 2 & 5 & 3 \\ 3 & 2 & 1 \\ 3 & 4 & 2 \end{pmatrix}$$

Вектор B:

$$\begin{pmatrix} 1 \\ 2 \\ -2 \end{pmatrix}$$

4.2.2 Результат

```
  /\_/\
 ( ^.^ )
  > ^ <
 Система имеет единственное решение!
Треугольная матрица:
  3.0000   2.0000   1.0000 |   2.0000
  0.0000   3.6667   2.3333 |  -0.3333
  0.0000   0.0000  -0.2727 |  -3.8182
```

```
Определитель матрицы: -2.9999999987
Вектор неизвестных:
X[0] =   2.0000
X[1] =  -9.0000
X[2] =  14.0000
Вектор невязок:
R[0] =   0.0000
R[1] =  -0.0000
R[2] =   0.0000
Проверка решения через numpy:
Решение (numpy): [ 2. -9. 14.]
Определитель (numpy): -3.000000000000001
  /\_/\
 ( ^.^ )
  > ^ <
 Ура! Мы справились!
```

4.3 Пример 3

4.3.1 Входные данные

Матрица A:

$$\begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix}$$

Вектор B:

$$\begin{pmatrix} 4 \\ 5 \end{pmatrix}$$

4.3.2 Результат

```
  /\_/\
 ( T.T )
  > ^ <
Ошибка: Система имеет бесконечно много решений.
```

5 Вывод

В ходе выполнения лабораторной работы я реализовал метод Гаусса с выбором главного элемента по столбцам для решения СЛУ на скриптовом языке bash. Программа выводит решение, определитель и вектор невязок. Реализована проверка решения с помощью библиотеки numpy на языке1 python.