# Линейная алгебра

Гань Чжаолун

6 декабрь, 2024, Москва, Россия

Российский Университет Дружбы Народов

# Цели и задачи работы

# Цель лабораторной работы

Основной целью работы является изучение возможностей специализированных пакетов Julia для выполнения и оценки эффективности операций над объектами линейной алгебры.

# Процесс выполнения лабораторной работы

# Используя Jupyter Lab, повторите примеры из раздела 4.2

Я повторю все задание 4.2 целиком

# Выполните задания для самостоятельной работы

## 4.4.1. Произведение векторов

1. Задайте вектор v. Умножьте вектор v скалярно сам на себя и сохраните результат в dot_v.
2. Умножьте v матрично на себя (внешнее произведение), присвоив результат переменной outer_v.

```
v = [3, 5, 8, 4]
dot_v = dot(v, v)
```

```
114
```

```
outer_v = v * v'
```

```
4×4 Matrix{Int64}:
  9  15  24  12
 15  25  40  20
 24  40  64  32
 12  20  32  16
```

1.Скалярное произведение (dot_v): Результатом является скаляр, который вычисляется как сумма произведений соответствующих элементов вектора на самих себя. В данном случае результат равен 114.

2.Внешнее произведение (outer_v): Результатом является матрица 4x4, каждый элемент которой представляет собой произведение соответствующих элементов вектора. Полученная матрица является симметричной.

Рисунок 1. Код и результат Задания 1

# Выполните задания для самостоятельной работы


1.Представление СЛАУ в матричной форме: СЛАУ с двумя или несколькими неизвестными можно записать в матричной форме, где матрица коэффициентов переменных (A) умножается на вектор неизвестных (x), что дает вектор значений правой части (b):

$$A * x = b$$

Здесь A — это матрица коэффициентов, а b — вектор правых частей.

2.Решение через оператор "левого деления" (\): В Julia для решения линейных уравнений используется оператор \, который находит вектор неизвестных x:

$$x = A \setminus b$$

Этот оператор находит решение, используя методы линейной алгебры, такие как метод Гаусса или LU-разложение.

Рисунок 2. Код и результат Задания 2

# Выполните задания для самостоятельной работы

### 4.4.3 Операции с матрицами

**Приведите приведенные ниже матрицы к диагональному виду**

```julia
function diagonal_matrices(matrix)
    # Проведем симметризацию матриц
    Asym = matrix + matrix'
    # Спектральное разложение симметризированной матрицы
    AsymEig = eigen(Asym)
    # в итоге приводим матрицу к диагональному виду
    return inv(AsymEig.vectors) * matrix * AsymEig.vectors
end
```

```
diagonal_matrices (generic function with 1 method)
```

```julia
matrix_1 = [1 -2; -2 1]
diagonal_matrices(matrix_1)
```

```
2×2 Matrix{Float64}:
 -1.0  0.0
  0.0  3.0
```

```julia
matrix_2 = [1 -2; -2 3]
diagonal_matrices(matrix_2)
```

```
2×2 Matrix{Float64}:
 -0.236068     4.44089e-16
  2.22045e-16  4.23607
```

```julia
matrix_3 = [1 -2 0; -2 1 2; 0 2 0]
diagonal_matrices(matrix_3)
```

```
3×3 Matrix{Float64}:
 -2.14134       3.55271e-15  -1.9984e-15
  3.38618e-15   0.515138      1.11022e-16
 -6.66134e-16  -4.44089e-16   3.6262
```

1.Симметризация: Первым шагом симметризуем матрицу, чтобы работать с более удобной в вычислениях версией, так как симметричные матрицы имеют вещественные собственные значения.

2.Спектральное разложение: Вычисляем собственные значения и векторы симметризованной матрицы, что позволяет привести исходную матрицу к диагональному виду.

3.Диагонализация: Используем собственные векторы для преобразования матрицы в диагональный вид, что упрощает ее дальнейший анализ и вычисления.

Рисунок 3.1. Код и результат Задания 3-1

# Выполните задания для самостоятельной работы

**Вычислим матрицы**

```
([1 -2; -2 1])^10
```

```
2×2 Matrix{Int64}:
  29525  -29524
 -29524   29525
```

```
sqrt([5 -2; -2 5])
```

```
2×2 Matrix{Float64}:
  2.1889   -0.45685
 -0.45685   2.1889
```

```
([1 -2; -2 1])^(1/3)
```

```
2×2 Symmetric{ComplexF64, Matrix{ComplexF64}}:
  0.971125+0.433013im  -0.471125+0.433013im
 -0.471125+0.433013im   0.971125+0.433013im
```

```
sqrt([1 2; 2 3])
```

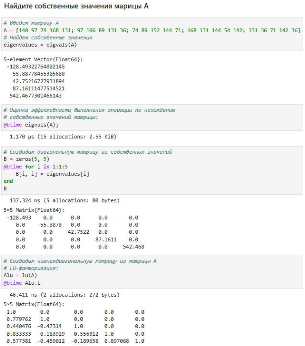
```
2×2 Matrix{ComplexF64}:
 0.568864+0.351578im  0.920442-0.217287im
 0.920442-0.217287im   1.48931+0.134291im
```

1.Возведение матрицы в степень: Использую оператор ^, чтобы возвести матрицу в нужную степень. Это полезно, когда необходимо работать с многократными линейными преобразованиями.

2.Извлечение корня: Применяю функцию sqrt() для извлечения квадратного корня из матрицы, что работает с симметричными положительно определёнными матрицами. Для кубического корня применяю возведение в степень 1/3.

Рисунок 3.2. Код и результат Задания 3-2

# Выполните задания для самостоятельной работы



Найдите собственные значения марицы А

```
# Введем матрицу А
A = [140 97 74 168 131; 97 106 89 131 36; 74 89 152 144 71; 168 131 144 54 142; 131 36 71 142 36]
# Найдем собственные значения
eigenvalues = eigvals(A)
```

```
5-element Vector{Float64}:
 -128.49322764802145
  -55.88778455305688
   42.75216727931894
   87.16111477514521
  542.4677301466143
```

```
# Оценка эффективности выполнения операции по нахождению
# собственных значений матрицы:
@btime eigvals(A);
```

```
  1.170 μs (15 allocations: 2.55 KiB)
```

```
# Создадим диагональную матрицу из собственных значений
B = zeros(5, 5)
@btime for i in 1:1:5
    B[i, i] = eigenvalues[i]
end
B
```

```
  137.324 ns (5 allocations: 80 bytes)
5×5 Matrix{Float64}:
 -128.493    0.0       0.0       0.0       0.0
    0.0    -55.8878    0.0       0.0       0.0
    0.0      0.0      42.7522    0.0       0.0
    0.0      0.0       0.0      87.1611    0.0
    0.0      0.0       0.0       0.0     542.468
```

```
# Создадим нижнеждиагональную матрицу из матрицы А
# LU-факторизация:
Alu = lu(A)
@btime Alu.L
```

```
  46.411 ns (2 allocations: 272 bytes)
5×5 Matrix{Float64}:
 1.0        0.0        0.0        0.0       0.0
 0.779762   1.0        0.0        0.0       0.0
 0.440476  -0.47314    1.0        0.0       0.0
 0.833333   0.183929  -0.556312   1.0       0.0
 0.577381  -0.459012  -0.189658   0.897068  1.0
```

1.Собственные значения: Для нахождения собственных значений использую eigvals(). Это важный шаг для анализа структуры матрицы и её свойств.

2.Диагональная матрица: Создаю диагональную матрицу из собственных значений, чтобы легко анализировать свойства матрицы в диагонализованной форме.

3.Нижняя треугольная матрица: Использую LU-разложение для выделения нижней треугольной матрицы. Этот метод удобен для анализа, поскольку LU-разложение разбивает матрицу на произведение двух треугольных матриц, что упрощает дальнейшие вычисления.

Рисунок 3.3. Код и результат Задания 3-3

# Выполните задания для самостоятельной работы

### 4.4.4 Линейные модели экономики

```
function productive_matrix(matrix, size)
    ans=""
    # единичная матрица
    E = [1 0; 0 1]
    # зададим любые неотрицательные числа
    Y = rand(0:1000, size)
    # По формуле вычислим x - A*x = y
    S = E - matrix
    # найдем значения x
    X = S\Y
    # теперь проверим есть ли среди x отрциательное число
    for i in 1:1:size
        if X[i] < 0
            ans = "Матрица непродуктивная"
            break
        else
            ans = "Матрица продуктивная"
        end
    end
    return ans
end
```

```
productive_matrix (generic function with 1 method)
```

```
matrix_1 = [1 2; 3 4]
productive_matrix(matrix_1, 2)
```

```
"Матрица непродуктивная"
```

```
matrix_2 = ([1 2; 3 4])*(1/2)
productive_matrix(matrix_2, 2)
```

```
"Матрица непродуктивная"
```

```
matrix_3 = ([1 2; 3 4])*(1/10)
productive_matrix(matrix_3, 2)
```

```
"Матрица продуктивная"
```

1.Проверка продуктивности через систему уравнений: Сначала формирую матрицу вида E - A и решаю систему линейных уравнений с случайной неотрицательной правой частью. Если среди элементов решения есть отрицательные, матрица считается непродуктивной.

2.Обратная матрица для анализа знаков: Также использую метод нахождения обратной матрицы E - A и анализирую все её элементы на наличие отрицательных значений. Отрицательные элементы свидетельствуют о непродуктивности матрицы.

Рисунок 4.1. Код и результат Задания 4-1

# Выполните задания для самостоятельной работы

```
function productive_matrix_2(matrix, size)
    # единичная матрица
    ans = ""
    E = [1 0; 0 1]
    matrix_new = E - matrix
    inv_matrix_new = inv(matrix_new)
    for i in 1:1:size
        for j in 1:1:size
            if inv_matrix_new[i, j] < 0
                ans = "Матрица непродуктивная"
                break
            else
                ans = "Матрица продуктивная"
            end
        end
    end
    return ans
end
```

```
productive_matrix_2 (generic function with 1 method)
```

```
matrix_1 = [1 2; 3 1]
productive_matrix_2(matrix_1, 2)
```

```
"Матрица непродуктивная"
```

```
matrix_2 = ([1 2; 3 1])*(1/2)
productive_matrix_2(matrix_2, 2)
```

```
"Матрица непродуктивная"
```

```
matrix_3 = ([1 2; 3 1])*(1/10)
productive_matrix_2(matrix_3, 2)
```

```
"Матрица продуктивная"
```

1.Формирование обратной матрицы: Сначала вычисляю обратную матрицу для (E - A). Продуктивность матрицы A определяется тем, что все элементы обратной матрицы должны быть неотрицательными.

2.Проверка на неотрицательность: Анализирую все элементы обратной матрицы. Если среди них встречаются отрицательные значения, матрица считается непродуктивной. Если все элементы неотрицательные, то матрица является продуктивной.

Рисунок 4.2. Код и результат Задания 4-2

# Выполните задания для самостоятельной работы

```
function productive_matrix_3(matrix, size)
    ans=""
    # найдем собственные значения переданной матрицы
    eigenvalues = eigvals(matrix)
    for i in 1:1:size
        if abs(eigenvalues[i]) > 1
            ans = "Матрица непродуктивная"
            break
        else
            ans = "Матрица продуктивная"
        end
    end
    return ans
end
```

```
productive_matrix_3 (generic function with 1 method)
```

```
matrix_1 = [1 2; 3 1]
productive_matrix_3(matrix_1, 2)
```

```
"Матрица непродуктивная"
```

```
matrix_2 = ([1 2; 3 1])*(1/2)
productive_matrix_3(matrix_2, 2)
```

```
"Матрица непродуктивная"
```

```
matrix_3 = ([1 2; 3 1])*(1/10)
productive_matrix_3(matrix_3, 2)
```

```
"Матрица продуктивная"
```

```
matrix_4 = [0.1 0.2 0.3; 0 0.1 0.2; 0 0.1 0.3]
productive_matrix_3(matrix_4, 2)
```

```
"Матрица продуктивная"
```

1.Спектральный критерий: Сначала нахожу собственные значения матрицы. Продуктивность матрицы A определяется тем, что все её собственные значения по модулю должны быть меньше 1.

2.Проверка собственных значений: Анализирую модуль каждого собственного значения. Если хотя бы одно значение больше или равно 1, то матрица считается непродуктивной. Если все значения меньше 1, то матрица продуктивная.

Рисунок 4.3. Код и результат Задания 4-3

# Выводы по проделанной работе

## Вывод

Я изучил возможности специализированных пакетов Julia для выполнения и оценки эффективности операций над объектами линейной алгебры. А также изучил основные методы факторизации объектов.