

# Лабораторная работа № 4.

## Линейная алгебра

Адабор Кристофер Твум

1032225824

НКНБД-01-22

## Цель работы

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

# Задание(повторяем примеры из раздела)

## Поэлементные операции над многомерными массивами

```
# Массив 4x3 со случайными целыми числами (от 1 до 20):  
a = rand(1:20, (4,3))  
# Поэлементная сумма:  
sum(a)  
# Поэлементная сумма по столбцам:  
sum(a,dims=1)  
# Поэлементная сумма по строкам:  
sum(a,dims=2)  
# Поэлементное произведение:  
prod(a)  
# Поэлементное произведение по столбцам:  
prod(a,dims=1)  
# Поэлементное произведение по строкам:  
prod(a,dims=2)
```

4x1 Matrix{Int64}:

40
1200
2176
216

```
# Подключение пакета Statistics:  
import Pkg  
Pkg.add("Statistics")  
using Statistics  
# Вычисление среднего значения массива:  
mean(a)  
# Среднее по столбцам:  
mean(a,dims=1)  
# Среднее по строкам:  
  
mean(a,dims=2)  
  
Resolving package versions...  
No Changes to 'C:\Users\adiiks\.julia\environments\v1.9\Project.toml'  
No Changes to 'C:\Users\adiiks\.julia\environments\v1.9\Manifest.toml'  
4x1 Matrix{Float64}:  
 5.0  
12.33333333333334  
13.666666666666666  
7.666666666666667
```

# Транспонирование, след, ранг, определитель и инверсия матрицы

```
# Подключение пакета LinearAlgebra:  
import Pkg  
Pkg.add("LinearAlgebra")  
using LinearAlgebra  
# Массив 4x4 со случайными целыми числами (от 1 до 20):  
b = rand(1:20,(4,4))  
# Транспонирование:  
transpose(b)  
# След матрицы (сумма диагональных элементов):  
tr(b)  
# Извлечение диагональных элементов как массив:  
diag(b)  
# Ранг матрицы:  
rank(b)  
# Инверсия матрицы (определение обратной матрицы):  
inv(b)  
# Определитель матрицы:  
det(b)  
# ПсевдоДобратная функция для прямоугольных матриц:  
pinv(a)
```

```
Resolving package versions...
```

```
Updating `C:\Users\adiks\.julia\environments\v1.9\Project.toml'  
[37e2e46d] + LinearAlgebra  
No Changes to `C:\Users\adiks\.julia\environments\v1.9\Manifest.toml'  
3x4 Matrix{Float64}:  
0.0120272  0.150642  -0.103743  -0.0217696  
-0.0660249 -0.199137  0.189898  0.0683751  
0.119144   0.0689872 -0.0554635 -0.0463341
```

# Вычисление нормы векторов и матриц, повороты, вращения

```
# Создание вектора X:  
X = [2, 4, -5]  
# Вычисление евклидовой нормы:  
norm(X)  
# Вычисление p-нормы:
```

```
p = 1  
norm(X,p)
```

11.0

```
# Рассстояние между двумя векторами X и Y:  
X = [2, 4, -5];  
Y = [1, -1, 3];  
norm(X-Y)
```

9.486832980585138

```
In [11]: # Угол между двумя векторами:  
acos((transpose(X)*Y)/(norm(X)*norm(Y)))
```

```
Out[11]: 2.4404307889469252
```

Вычисление нормы для двумерной матрицы:

```
In [12]: # Создание матрицы:  
d = [5 -4 2 ; -1 2 3; -2 1 0]  
# Вычисление Евклидовой нормы:  
opnorm(d)  
# Вычисление p-нормы:  
p=1  
opnorm(d,p)  
# Поворот на 180 градусов:  
rot180(d)  
# Переворачивание строк:  
reverse(d,dims=1)  
# Переворачивание столбцов  
reverse(d,dims=2)
```

```
Out[12]: 3x3 Matrix{Int64}:
```

2	-4	5
3	2	-1
0	1	-2

# Матричное умножение, единичная матрица, скалярное произведение

```
: # Матрица 2x3 со случайными целыми значениями от 1 до 10:  
A = rand(1:10,(2,3))  
# Матрица 3x4 со случайными целыми значениями от 1 до 10:  
B = rand(1:10,(3,4))  
# Произведение матриц A и B:  
A*B  
# Единичная матрица 3x3:
```

```
Matrix{Int}(I, 3, 3)  
# Скалярное произведение векторов X и Y:  
X = [2, 4, -5]  
Y = [1,-1,3]  
dot(X,Y)  
# тоже скалярное произведение:  
X*Y  
  
println("Уденичная матрица ", Matrix{Int}(I, 3, 3))  
println("Скалярное произведение ", dot(X,Y))
```

Уденичная матрица [1 0 0; 0 1 0; 0 0 1]

Скалярное произведение -17

## Факторизация. Специальные матричные структуры

```
|: # Задаём квадратную матрицу 3х3 со случайными значениями:  
A = rand(3, 3)  
# Задаём единичный вектор:  
x = fill(1.0, 3)  
# Задаём вектор b:  
b = A*x  
# Решение исходного уравнения получаем с помощью функции l  
# (убеждаемся, что x - единичный вектор):  
A\b  
  
|: 3-element Vector{Float64}:  
 1.000000000000002  
 1.0  
 1.0  
  
|: # LU-факторизация:  
Alu = lu(A)  
  
|: LU{Float64, Matrix{Float64}, Vector{Int64}}  
L factor:  
3x3 Matrix{Float64}:  
 1.0      0.0      0.0  
 0.137732  1.0      0.0  
 0.267182  0.925564  1.0  
U factor:  
3x3 Matrix{Float64}:  
 0.591678  0.887178  0.486095  
 0.0       0.348887  0.743889  
 0.0       0.0       0.144943
```

```
# Матрица перестановок:  
Alu.P  
# Вектор перестановок:  
Alu.p  
# Матрица L:  
Alu.L  
# Матрица U:  
Alu.U
```

```
# Детерминант матрицы A:  
det(A)  
# Детерминант матрицы A через объект факторизации:  
det(Alu)  
  
0.029920458614848326
```

```
# QR-факторизация:  
Aqr = qr(A)
```

```
LinearAlgebra.QRCompactWY{Float64, Matrix{Float64}, Matrix{Float64}}  
Q factor:  
3x3 LinearAlgebra.QRCompactWYQ{Float64, Matrix{Float64}, Matrix{Float64}}:  
-0.95767   0.269196  -0.181992  
-0.131902  -0.725259  -0.675723  
-0.255872  -0.633666  0.730066  
R factor:  
3x3 Matrix{Float64}:  
-0.617831  -0.971581  -0.818961  
 0.0         -0.457655  -1.06765  
 0.0          0.0        0.105818
```

```
3x3 Matrix{Float64}:  
0.591678  0.807178  0.486095  
0.0        0.348887  0.743889  
0.0        0.0        0.144943
```

```
# Решение СЛАУ через матрицу A:  
A\b  
# Решение СЛАУ через объект факторизации:  
Alu\b
```

```
3-element Vector{Float64}:  
1.000000000000002  
1.0  
1.0
```

```
: # Матрица Q:  
Aqr.Q  
# Матрица R:  
Aqr.R  
# Проверка, что матрица Q - ортогональная:  
Aqr.Q %*% Aqr.Q
```

```
: 3x3 Matrix{Float64}:  
 1.0          0.0      -5.55112e-17  
 5.55112e-17  1.0       1.11022e-16  
 0.0          1.11022e-16  1.0
```

```
: # Симметризация матрицы A:  
Asym = A + A'  
# Спектральное разложение симметризованной матрицы:  
AsymEig = eigen(Asym)  
# Собственные значения:  
AsymEig$values  
# Собственные векторы:
```

```
AsymEig.vectors
# Проверяем, что получится единичная матрица:
inv(AsymEig)*Asym
```

```
: 3x3 Matrix{Float64}:
 1.0      -8.65974e-15   6.21725e-15
 2.22045e-15    1.0      -1.06581e-14
 -3.33067e-16  -6.66134e-15   1.0
```

```
# Добавление шума:
Asym_noisy = copy(Asym)
Asym_noisy[1,2] += 5eps()
# Проверка, является ли матрица симметричной:
issymmetric(Asym_noisy)
```

```
: false
```

```
# Явно указываем, что матрица является симметричной
Asym_explicit = Symmetric(Asym_noisy)
```

```
: 3x3 Symmetric{Float64, Matrix{Float64}}:
 1.18336  0.888672  0.644181
 0.888672  0.920123  1.34942
 0.644181  1.34942   1.92667
```

```
import Pkg
Pkg.add("BenchmarkTools")
using BenchmarkTools
# Оценка эффективности выполнения операции по нахождению
# собственных значений симметризованной матрицы:
@btime eigvals(Asym);
# Оценка эффективности выполнения операции по нахождению
# собственных значений зашумлённой матрицы:
@btime eigvals(Asym_noisy);
# Оценка эффективности выполнения операции по нахождению
# собственных значений зашумлённой матрицы,
# для которой явно указано, что она симметричная:
@btime eigvals(Asym_explicit);
```

```
Resolving package versions...
Installed BenchmarkTools - v1.4.0
Updating `C:\Users\adiks\julia\environments\v1.9\Project.toml`
[6e4b80f9] + BenchmarkTools v1.4.0
Updating `C:\Users\adiks\julia\environments\v1.9\Manifest.toml`
[6e4b80f9] + BenchmarkTools v1.4.0
[9abbd945] + Profile
>recompiling project...
✓ BenchmarkTools
1 dependency successfully precompiled in 5 seconds. 204 already precompiled.
1.830 μs (10 allocations: 1.69 KiB)
2.367 μs (12 allocations: 1.64 KiB)
1.670 μs (10 allocations: 1.69 KiB)

# Трёхдиагональная матрица 1000000 x 1000000:
n = 1000000;
A = SymTridiagonal(randn(n), randn(n-1))
```

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

417.457 ms (17 allocations: 183.11 MiB)
]: 6.919247259252579
```

## Общая линейная алгебра

```
]: # Матрица с рациональными элементами:
Arational = Matrix{Rational{BigInt}}(rand(1:10, 3, 3))/10
# Единичный вектор:
x = fill(1, 3)
# Задаём вектор b:
b = Arational*x
# Решение исходного уравнения получаем с помощью функции \
# (убеждаемся, что x - единичный вектор):
Arational\b
# LU-разложение:
lu(Arational)

]: LU{Rational{BigInt}, Matrix{Rational{BigInt}}, Vector{Int64}}
L factor:
3x3 Matrix{Rational{BigInt}}:
 1//1  0//1  0//1
 1//4  1//1  0//1
 1//8  27//46  1//1
U factor:
3x3 Matrix{Rational{BigInt}}:
 4//5  1//2  2//5
 0//1  23//40  7//10
 0//1  0//1  16//115
```

# Задания для самостоятельного выполнения

## Произведение векторов

```
# 1. Скалярное умножение вектора на самого себя
v = [1, 2, 3] # Пример вектора, замените на ваш вектор

dot_v = dot(v, v) # Результат скалярного умножения

# 2. Внешнее произведение вектора на самого себя
outer_v = v * v' # Результат внешнего произведения

# Вывод результатов
println("Скалярное произведение: $dot_v")
println("Внешнее произведение: $outer_v")
```

```
Скалярное произведение: 14
Внешнее произведение: [1 2 3; 2 4 6; 3 6 9]
```

## Системы линейных уравнений

```
: using LinearAlgebra

# Функция для решения СЛАУ и обработки исключений
function solve_system(A, b)
    try
        return A \ b
    catch e
        if isa(e, SingularException)
            return "Система не имеет единственного решения (вырожденная или несовместная)"
        else
            throw(e)
        end
    end
end

# Решение СЛАУ
```

```
# a)
A1 = [1 1; 1 -1]
b1 = [2; 3]
x1 = solve_system(A1, b1)

# b)
A2 = [1 1; 2 2]
b2 = [2; 4]
x2 = solve_system(A2, b2)

# c)
A3 = [1 1; 2 2]
b3 = [2; 5]
x3 = solve_system(A3, b3)

# d)
A4 = [1 1; 2 2; 3 3]
b4 = [1; 2; 3]
x4 = solve_system(A4, b4)

# e)
A5 = [1 1; 2 1; 1 -1]
b5 = [2; 1; 3]
x5 = solve_system(A5, b5)

# f)
A6 = [1 1; 2 1; 3 2]
b6 = [2; 1; 3]
x6 = solve_system(A6, b6)

# Вывод результатов
println("Решение а): $x1")
println("Решение б): $x2")
```

```
println("Решение с): $x3")
println("Решение д): $x4")
println("Решение е): $x5")
println("Решение ф): $x6")
```

Решение а): [2.5, -0.5]

Решение б): Система не имеет единственного решения (вырожденная или несовместная)

Решение с): Система не имеет единственного решения (вырожденная или несовместная)

Решение д): [0.4999999999999999, 0.5]

Решение е): [1.500000000000004, -0.999999999999997]

Решение ф): [-0.999999999999989, 2.999999999999982]

# Операции с матрицами

```
: using LinearAlgebra
```

```
# 1. Приведение матриц к диагональному виду
```

```
# a)  
A1 = [1 -2; -2 1]  
D1 = Diagonal(eigvals(A1))  
  
# b)  
A2 = [1 -2; -2 3]  
D2 = Diagonal(eigvals(A2))  
  
# c)  
A3 = [1 -2 0; -2 1 2; 0 2 0]  
D3 = Diagonal(eigvals(A3))  
  
# 2. Вычисление степеней и корней матриц
```

```
# a)  
pow_A1 = A1^10
```

```
# b)  
sqrt_A2 = sqrt(A2)
```

```
# c)  
cbrt_A1 = cbrt.(A1)
```

```
# d)  
A4 = [1 2; 2 3]  
sqrt_A4 = sqrt(A4)
```

```
A5 = [140 97 74 168 131; 97 106 89 131 36; 74 89 152 144 71; 168 131 144 54 142; 13  
eigenvalues_A5 = eigvals(A5)  
D5 = Diagonal(eigenvalues_A5)  
  
# Нижнедиагональная матрица из A  
lower_A5 = tril(A5)  
  
# Вывод результатов
```

```
println("Диагональный вид а): $D1")
println("Диагональный вид б): $D2")
println("Диагональный вид с): $D3")
println("A1 в 10 степени: $pow_A1")
println("Квадратный корень из A2: $sqrt_A2")
println("Кубический корень из A1: $cbrt_A1")
println("Квадратный корень из A4: $sqrt_A4")
println("Собственные значения A5: $eigenvalues_A5")
println("Диагональная матрица из собственных значений A5: $D5")
println("Нижнедиагональная матрица из A5: $lower_A5")
```

```
Диагональный вид а): [-1.0 0.0; 0.0 3.0]
Диагональный вид б): [-0.2360679774997897 0.0; 0.0 4.23606797749979]
Диагональный вид с): [-2.1413361156553643 0.0 0.0; 0.0 0.51513804712807 0.0; 0.0 0.0
3.6261980685272945]
A1 в 10 степени: [29525 -29524; -29524 29525]
Квадратный корень из A2: ComplexF64[0.5688644810057828 + 0.35157758425414287im -0.92
04420652599258 + 0.2172868967516401im; -0.9204420652599258 + 0.21728689675164012im 1
,489306546265709 + 0.1342906875025027im]
Кубический корень из A1: [1.0 -1.2599210498948732; -1.2599210498948732 1.0]
Квадратный корень из A4: ComplexF64[0.5688644810057828 + 0.35157758425414287im 0.920
4420652599258 - 0.2172868967516401im; 0.9204420652599258 - 0.21728689675164012im 1.4
89306546265709 + 0.1342906875025027im]
Собственные значения A5: [-128.49322764802145, -55.887784553056875, 42.7521672793189
, 87.16111477514521, 542.4677301466143]
Диагональная матрица из собственных значений A5: [-128.49322764802145 0.0 0.0 0.0 0.
0; 0.0 -55.887784553056875 0.0 0.0 0.0; 0.0 0.0 42.7521672793189 0.0 0.0; 0.0 0.0 0.
0 87.16111477514521 0.0; 0.0 0.0 0.0 0.0 542.4677301466143]
Нижнедиагональная матрица из A5: [140 0 0 0 0; 97 106 0 0 0; 74 89 152 0 0; 168 131
144 54 0; 131 36 71 142 36]
```

# Линейные модели экономики

```
: using LinearAlgebra

# Функция для проверки производности матрицы
function is_productive(A)
    E = Matrix{Float64}(I, size(A))
    inv_E_A = inv(E - A)
    return all(inv_E_A .≥ 0)
end

# Функция для проверки спектрального критерия производности
function is_spectral_productive(A)
    eigenvalues = abs.(eigvals(A))
    return all(eigenvalues .< 1)
end

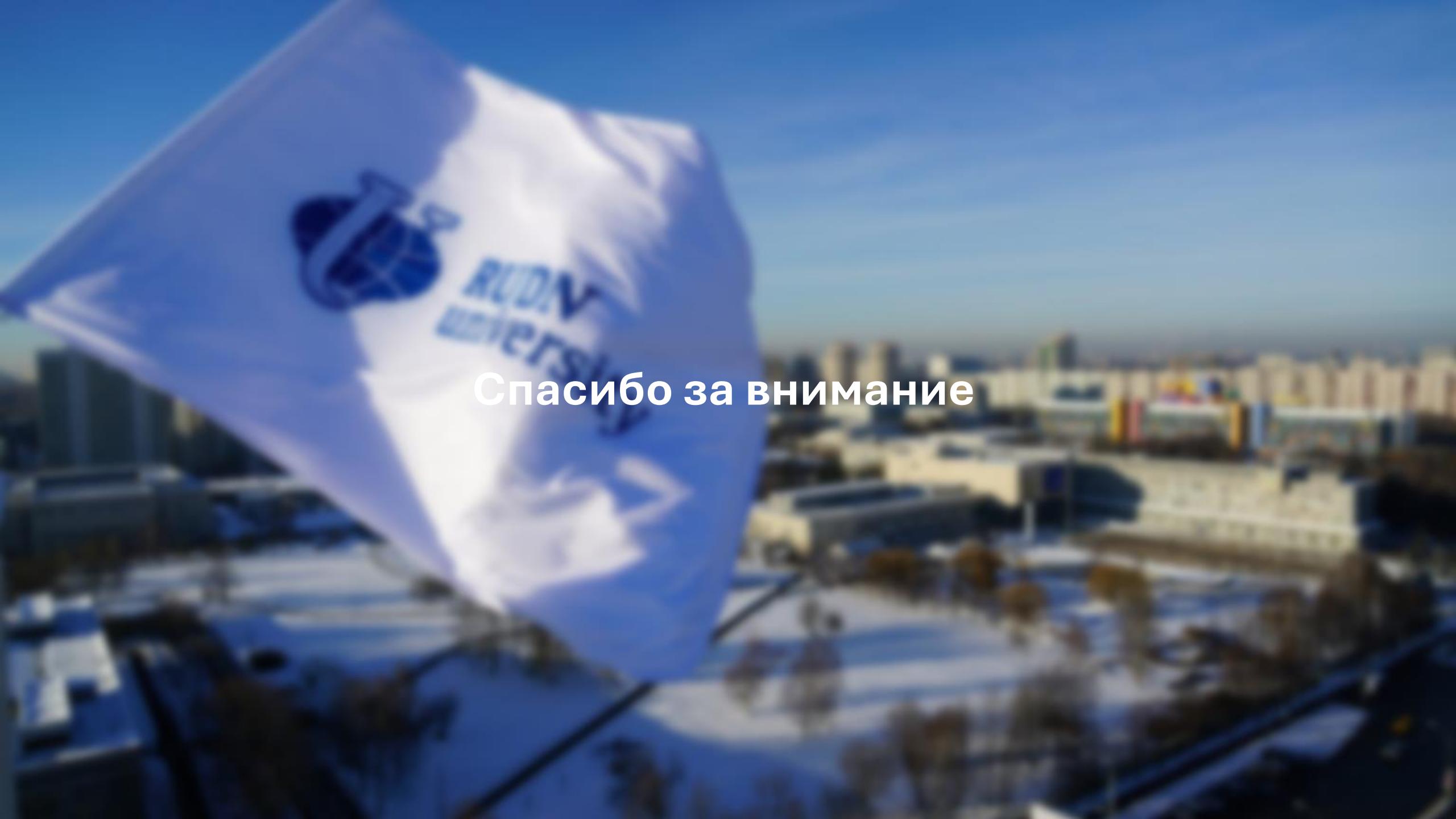
# Матрицы для проверки
A1 = [1 2; 3 4]
A2 = 1/2 * A1
A3 = 1/10 * A1
A4 = [1 2; 3 1]
A5 = 1/2 * A4
A6 = 1/10 * A4
```

```
A7 = [0.1 0.2 0.3; 0 0.1 0.2; 0 0.1 0.3]

# Проверка продуктивности
println("Продуктивность A1: ${is_productive(A1)}")
println("Продуктивность A2: ${is_productive(A2)}")
println("Продуктивность A3: ${is_productive(A3)}")
println("Продуктивность A4: ${is_productive(A4)}")
println("Продуктивность A5: ${is_productive(A5)}")
println("Продуктивность A6: ${is_productive(A6)}")

# Проверка спектрального критерия продуктивности
println("Спектральная продуктивность A1: ${is_spectral_productive(A1)}")
println("Спектральная продуктивность A2: ${is_spectral_productive(A2)}")
println("Спектральная продуктивность A3: ${is_spectral_productive(A3)}")
println("Спектральная продуктивность A4: ${is_spectral_productive(A4)}")
println("Спектральная продуктивность A5: ${is_spectral_productive(A5)}")
println("Спектральная продуктивность A6: ${is_spectral_productive(A6)}")
println("Спектральная продуктивность A7: ${is_spectral_productive(A7)})
```

```
Продуктивность A1: false
Продуктивность A2: false
Продуктивность A3: true
Продуктивность A4: false
Продуктивность A5: false
Продуктивность A6: true
Спектральная продуктивность A1: false
Спектральная продуктивность A2: false
Спектральная продуктивность A3: true
Спектральная продуктивность A4: false
Спектральная продуктивность A5: false
Спектральная продуктивность A6: true
Спектральная продуктивность A7: true
```



Спасибо за внимание