РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖ|БЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра теории вероятностей и кибербезопасности

Лабораторная работа № 4 | Линейная алгебра

Студент: Абд эль хай Мохамад
Номер: 1032215163
Група: НПИбд-01-21

Москва 2025

Оглавление

Цель работы

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

Поэлементные операции над многомерными массивами

In [99]:
```
a = rand(1:9,(4,3))
```

Out[99]:
```
4×3 Matrix{Int64}:
 8  4  7
 1  3  5
 8  8  1
 8  6  1
```

In [100…
```
println("Поэлементная сумма ", sum(a))
println("Поэлементная сумма по столбцам ",sum(a, dims=1))
println("Поэлементная сумма по строкам ",sum(a, dims=2))
```

```
Поэлементная сумма 60
Поэлементная сумма по столбцам [25 21 14]
Поэлементная сумма по строкам [19; 9; 17; 15;;]
```

In [101…
```
println("Поэлементное произведение ",prod(a))
println("Поэлементное произведение по столбцам ",prod(a, dims=1))
println("Поэлементное произведение по строкам ",prod(a, dims=2))
```

```
Поэлементное произведение 10321920
Поэлементное произведение по столбцам [512 576 35]
Поэлементное произведение по строкам [224; 15; 64; 48;;]
```

In [1]:
```
# Подключение пакета Statistics:
import Pkg
Pkg.add("Statistics")
using Statistics
```

```
  Resolving package versions...
 No Changes to `~/.julia/environments/v1.11/Project.toml`
 No Changes to `~/.julia/environments/v1.11/Manifest.toml`
```

In [9]:
```
println("Вычисление среднего значения массива ", mean(a))
println("Среднее по столбцам ", mean(a,dims=1))
println("Среднее по строкам ", mean(a,dims=2))
```

```
Вычисление среднего значения массива 6.0
Среднее по столбцам [5.0 7.75 5.25]
Среднее по строкам [6.666666666666667; 5.666666666666667; 6.666666666666667;
5.0;;]
```

Транспонирование, след, ранг, определитель и инверсия матрицы

In [11]:
```
import Pkg
Pkg.add("LinearAlgebra")
using LinearAlgebra
```

```
  Resolving package versions...
   Updating `~/.julia/environments/v1.11/Project.toml`
 [37e2e46d] + LinearAlgebra v1.11.0
 No Changes to `~/.julia/environments/v1.11/Manifest.toml`
```

In [12]:
```
b = rand(1:20,(4,4))
```

Out[12]:
```
4×4 Matrix{Int64}:
  9  10   3   6
 13   1  12  19
 10   5   7  10
  7   3   9  16
```

In [13]:
```
println("Транспонированная матрица = ", transpose(b))
println("След матрицы (сумма диагональных элементов) = ", tr(b))
println("Извлечение диагональных элементов как массив = ", diag(b))
```

```
Транспонированная матрица = [9 13 10 7; 10 1 5 3; 3 12 7 9; 6 19 10 16]
След матрицы (сумма диагональных элементов) = 33
Извлечение диагональных элементов как массив = [9, 1, 7, 16]
```

In [14]:
```
println("Ранг матрицы = ", rank(b))
```

```
Ранг матрицы = 4
```

In [15]:
```
println("Инверсия матрицы = ")
inv(b)
```

```
Инверсия матрицы =
```

Out[15]:
```
4×4 Matrix{Float64}:
  0.148962   0.261294  -0.208791  -0.235653
 -0.018315  -0.212454   0.197802   0.135531
 -0.567766  -0.586081   1.13187    0.201465
  0.257631   0.255189  -0.582418   0.026862
```

In [16]:
```
println("Определитель матрицы = ", det(b))
```

```
Определитель матрицы = -818.9999999999985
```

In [17]:
```
println("Псевдобратная функция для прямоугольных матриц = ")
pinv(a)
```

```
Псевдобратная функция для прямоугольных матриц =
```

Out[17]:
```
3×4 Matrix{Float64}:
  0.255319   -0.0795073   0.161254  -0.398656
 -0.0673759   0.135747   -0.215068   0.289411
 -0.0957447  -0.0776409   0.211459   0.000373274
```

Вычисление нормы векторов и матриц, повороты, вращения

In [21]:
```
# Создание вектора X
X = [2,4,-5]
```

Out[21]:
```
3-element Vector{Int64}:
  2
  4
 -5
```

In [22]:
```
println("Евклидова норма вектора X = ", norm(X))
```

```
Евклидова норма вектора X = 6.708203932499369
```

In [23]:
```
p = 1
println("p-норма вектора X = ", norm(X, p))
```

```
p-норма вектора X = 11.0
```

In [24]:
```
X = [2, 4, -5];
Y = [1,-1,3];
println("Расстояние между векторами X и Y = ", norm(X-Y))
println("Расстояние по базовому определению(проверка) = ", sqrt(sum((X-Y).^2
```

```
Расстояние между векторами X и Y = 9.486832980505138
Расстояние по базовому определению(проверка) = 9.486832980505138
```

Матричное умножение, единичная матрица, скалярное произведение

In [28]:
```
# 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))

A*B
```

Out[28]:
```
2×4 Matrix{Int64}:
 61  17  43   81
 93  58  32  127
```

In [32]:
```
# Единичная матрица 3x3
Matrix{Int}(I,3,3)
```

Out[32]:
```
3×3 Matrix{Int64}:
 1  0  0
 0  1  0
 0  0  1
```

In [33]:
```
# Скалярное произведение векторов X и Y:
X = [2, 4, -5]
Y = [1,-1,3]
dot(X,Y)
```

Out[33]:
```
-17
```

Факторизация. Специальные матричные структуры

In [34]:
```
# Задаём квадратную матрицу 3x3 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 \
# (убеждаемся, что x - единичный вектор):
A\b
```

Out[34]:
```
3-element Vector{Float64}:
 1.0000000000000024
 0.9999999999999974
 0.9999999999999994
```

In [35]:
```
# LU-факторизация:
Alu = lu(A)
```

Out[35]:
```
LU{Float64, Matrix{Float64}, Vector{Int64}}
L factor:
3×3 Matrix{Float64}:
 1.0       0.0       0.0
 0.983686  1.0       0.0
 0.6667    0.323383  1.0
U factor:
3×3 Matrix{Float64}:
 0.614859  0.369507   0.822136
 0.0       0.110925  -0.271536
 0.0       0.0       -0.100201
```

In [36]:
```
println("Матрица перестановок = ", Alu.P)
println("Вектор перестановок = ", Alu.p)
```

```
Матрица перестановок = [1.0 0.0 0.0; 0.0 0.0 1.0; 0.0 1.0 0.0]
Вектор перестановок = [1, 3, 2]
```

Общая линейная алгебра

Arational = Matrix{Rational{BigInt}}(rand(1:10,3,3))/10

In [39]:
```
# Единичный вектор:
x = fill(1, 3)
# Задаём вектор b
b = Arational*x
```

Out[39]:
```
3-element Vector{Rational{BigInt}}:
 21//10
 13//10
  7//10
```

In [40]:
```
# Решение исходного уравнения получаем с помощью функции \
# (убеждаемся, что x - единичный вектор):
Arational\b
```

Out[40]:
```
3-element Vector{Rational{BigInt}}:
 1
 1
 1
```

In [41]:
```
# LU-разложение:
lu(Arational)
```

Out[41]:
```
LU{Rational{BigInt}, Matrix{Rational{BigInt}}, Vector{Int64}}
L factor:
3×3 Matrix{Rational{BigInt}}:
  1    0  0
 1//6  1  0
 1//2  0  1
U factor:
3×3 Matrix{Rational{BigInt}}:
 3//5   4//5    7//10
  0    -1//30  23//60
  0      0      1//4
```

Задания для самостоятельного выполнения

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.

In [44]:
```
V = [3,6,8,-11]
dotV = dot(V,V)
```

Out[44]:
```
230
```

In [45]:
```
outerV = V * V'
```

Out[45]:
```
4×4 Matrix{Int64}:
   9   18   24  -33
  18   36   48  -66
  24   48   64  -88
 -33  -66  -88  121
```

Системы линейных уравнений

In [65]:
```
# first task
A_1 = [1 1; 1 -1]
b_1 = [2; 3]
# Решение уравнения получаем с помощью функции \
try
    A_1\b_1
catch e
    println("матрица A является вырожденной")
end
```

Out[65]:
```
2-element Vector{Float64}:
  2.5
 -0.5
```

In [66]:
```
# second task
A_2 = [1 1; 2 2]
b_2 = [2; 4]
try
    A_2\b_2
catch e
    println("матрица A является вырожденной")
end
```

```
матрица A является вырожденной
```

In [67]:
```
# third task
A_3 = [1 1; 2 2]
b_3 = [2; 5]
try
    A_3\b_3
catch e
    println("матрица A является вырожденной")
end
```

```
матрица A является вырожденной
```

In [68]:
```
# fourth task
A_4 = [1 1; 2 2; 3 3]
b_4 = [1; 2; 3]
A_4\b_4
```

Out[68]:
```
2-element Vector{Float64}:
 0.4999999999999999
 0.5
```

In [69]:
```
# fifth task
A_5 = [1 1; 2 1; 1 -1]
b_5 = [2; 1; 3]
A_5\b_5
```

Out[69]:
```
2-element Vector{Float64}:
  1.5000000000000004
 -0.9999999999999997
```

In [72]:
```
# ninth task
A_9 = [1 1 1; 1 1 2; 2 2 3]
b_9 = [1; 0; 1]
try
    A_9\b_9
catch e
    println("матрица A является вырожденной")
end
```

```
матрица A является вырожденной
```

Операции с матрицами

In [73]:
```
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
```

Out[73]:
```
diagonal_matrices (generic function with 1 method)
```

In [74]:
```
matrix_1 = [1 -2; -2 1]
diagonal_matrices(matrix_1)
```

Out[74]:
```
2×2 Matrix{Float64}:
 -1.0  0.0
  0.0  3.0
```

In [75]:
```
matrix_2 = [1 -2; -2 3]
diagonal_matrices(matrix_2)
```

Out[75]:
```
2×2 Matrix{Float64}:
 -0.236068     4.44089e-16
  2.22045e-16  4.23607
```

In [76]:
```
matrix_3 = [1 -2 0; -2 1 2; 0 2 0]
diagonal_matrices(matrix_3)
```

Out[76]:
```
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
```

Вычислим матрицы

In [77]:
```
([1 -2; -2 1])^10
```

Out[77]:
```
2×2 Matrix{Int64}:
  29525  -29524
 -29524   29525
```

In [79]:
```
sqrt([5 -2; -2 5])
```

Out[79]:
```
2×2 Matrix{Float64}:
  2.1889   -0.45685
 -0.45685   2.1889
```

In [80]:
```
([1 -2; -2 1])^(1/3)
```

Out[80]:
```
2×2 Symmetric{ComplexF64, Matrix{ComplexF64}}:
  0.971125+0.433013im  -0.471125+0.433013im
 -0.471125+0.433013im   0.971125+0.433013im
```

In [81]:
```
sqrt([1 2; 2 3])
```

Out[81]:
```
2×2 Matrix{ComplexF64}:
 0.568864+0.351578im  0.920442-0.217287im
 0.920442-0.217287im   1.48931+0.134291im
```

Линейные модели экономики

In [86]:
```
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
```

Out[86]:
```
productive_matrix (generic function with 1 method)
```

In [87]:
```
matrix_2 = ([1 2; 3 4])*(1/2)
productive_matrix(matrix_2, 2)
```

Out[87]:
```
"Матрица непродуктивная"
```

In [88]:
```
matrix_3 = ([1 2; 3 4])*(1/10)
productive_matrix(matrix_3, 2)
```

Out[88]:
```
"Матрица продуктивная"
```

In [89]:
```
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
```

Out[89]: productive_matrix_2 (generic function with 1 method)

In [90]:
```
matrix_1 = [1 2; 3 1]
productive_matrix_2(matrix_1, 2)
```

Out[90]: "Матрица непродуктивная"

In [91]:
```
matrix_2 = ([1 2; 3 1])*(1/2)
productive_matrix_2(matrix_2, 2)
```

Out[91]: "Матрица непродуктивная"

In [92]:
```
matrix_3 = ([1 2; 3 1])*(1/10)
productive_matrix_2(matrix_3, 2)
```

Out[92]: "Матрица продуктивная"

In [93]:
```
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
```

Out[93]: productive_matrix_3 (generic function with 1 method)

In [94]:
```
matrix_1 = [1 2; 3 1]
productive_matrix_3(matrix_1, 2)
```

Out[94]: "Матрица непродуктивная"

In [95]:
```
matrix_2 = ([1 2; 3 1])*(1/2)
productive_matrix_3(matrix_2, 2)
```

Out[95]: "Матрица непродуктивная"

In [96]:
```
matrix_3 = ([1 2; 3 1])*(1/10)
productive_matrix_3(matrix_3, 2)
```

Out[96]: "Матрица продуктивная"

In [97]:
```
matrix_4 = [0.1 0.2 0.3; 0 0.1 0.2; 0 0.1 0.3]
productive_matrix_3(matrix_4, 2)
```

Out[97]: "Матрица продуктивная"

In [ ]:

In [ ]: