

Лабораторная работа № 4

Линейная алгебра

Шияпова Д.И.

25 октября 2025

Российский университет дружбы народов, Москва, Россия

- Шияпова Дарина Илдаровна
- Студентка
- Российский университет дружбы народов
- 1132226458@pfur.ru



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

Выполнение лабораторной работы

```
[ ] ▶ # Массив 4x3 со случайными целыми числами (от 1 до 20):  
a = rand(1:20,(4,3))
```

```
↵ 4x3 Matrix{Int64}:  
 5 19 12  
 5 17 13  
 9  2 15  
18  8 16
```

```
[ ] # Поэлементная сумма:  
sum(a)
```

```
139
```

```
[ ] # Поэлементная сумма по столбцам:  
sum(a,dims=1)
```

```
1x3 Matrix{Int64}:  
37 46 56
```

```
[ ] # Поэлементная сумма по строкам:  
sum(a,dims=2)
```

```
4x1 Matrix{Int64}:  
36  
35  
26  
42
```

Выполнение лабораторной работы

```
# Вычисление среднего значения массива:
```

```
mean(a)
```

```
11.583333333333334
```

```
# Среднее по столбцам:
```

```
mean(a,dims=1)
```

```
1x3 Matrix{Float64}:
```

```
 9.25  11.5  14.0
```

```
# Среднее по строкам:
```

```
mean(a,dims=2)
```

```
4x1 Matrix{Float64}:
```

```
12.0
```

```
11.666666666666666
```

```
 8.666666666666666
```

```
14.0
```

```
# Подключение пакета LinearAlgebra:
```

```
import Pkg
```

```
Pkg.add("LinearAlgebra")
```

```
using LinearAlgebra
```

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

```
No Changes to `~/julia/environments/v1.11/Project.toml`
```

```
No Changes to `~/julia/environments/v1.11/Manifest.toml`
```

```
# Массив 4x4 со случайными целыми числами (от 1 до 20):
```

```
b = rand(1:20, (4,4))
```

```
4x4 Matrix{Int64}:
```

```
10  3  2  1
```

```
18  7  9 14
```

```
 7  6  5  1
```

```
15 15  1 19
```

```
# Транспонирование:
```

```
transpose(b)
```

Выполнение лабораторной работы

Для вычисления нормы используется `LinearAlgebra.norm(x)`.

Создание вектора X:

`X = [2, 4, -5]`

3-element Vector{Int64}:

2
4
-5

Вычисление евклидовой нормы:

`norm(X)`

6.708203932499369

Вычисление p-нормы:

`p = 1`

`norm(X,p)`

11.0

Расстояние между двумя векторами X и Y:

`X = [2, 4, -5];`

`Y = [1, -1, 3];`

`norm(X-Y)`

9.486832980505138

Проверка по базовому определению:

`sqrt(sum((X-Y).^2))`

9.486832980505138

Угол между двумя векторами:

`acos((transpose(X)*Y)/(norm(X)*norm(Y)))`

2.4404307889469252

Выполнение лабораторной работы

```
# 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))
```

```
3x4 Matrix{Int64}:  
 9  9  1  2  
 8  5 10  3  
 3  6  2  8
```

```
# Произведение матриц A и B:  
A*B
```

```
2x4 Matrix{Int64}:  
 82 106 44 94  
 74  86 58 80
```

```
# Единичная матрица 3x3:  
Matrix{Int}(I, 3, 3)
```

```
3x3 Matrix{Int64}:  
 1  0  0  
 0  1  0  
 0  0  1
```

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

```
-17
```

```
# тоже скалярное произведение:  
X*Y
```

```
-17
```

Выполнение лабораторной работы

```
# Задаём квадратную матрицу 3x3 со случайными значениями:  
A = rand(3, 3)
```

```
3x3 Matrix{Float64}:  
 0.953052  0.256058  0.668292  
 0.9789   0.655134  0.119584  
 0.852723  0.369336  0.0593227
```

```
# Задаём единичный вектор:  
x = fill(1.0, 3)
```

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

```
# Задаём вектор b:  
b = A*x
```

```
3-element Vector{Float64}:  
 1.8774028946026402  
 1.7536181410124794  
 1.281381554509779
```

```
# Решение исходного уравнения получаем с помощью функции \  
# (убеждаемся, что x - единичный вектор):  
A\b
```

```
3-element Vector{Float64}:  
 1.0000000000000007  
 0.9999999999999989  
 0.9999999999999997
```



```
# LU-факторизация:  
Alu = lu(A)
```



```
LU{Float64, Matrix{Float64}, Vector{Int64}}  
L factor:  
3x3 Matrix{Float64}:  
 1.0      0.0      0.0  
 0.973595 1.0      0.0  
 0.871103 0.527412 1.0  
U factor:
```


Выполнение лабораторной работы

Собственные значения:

AsymEig.values

3-element Vector{Float64}:

```
-0.7741474992917929  
0.5583384242456924  
3.5508269837372874
```

#Собственные векторы:

AsymEig.vectors

3x3 Matrix{Float64}:

```
-0.530506  0.377973 -0.758749  
0.117394 -0.853703 -0.507355  
0.839513  0.358228 -0.408523
```

Проверяем, что получится единичная матрица:

inv(AsymEig)*Asym

3x3 Matrix{Float64}:

```
1.0      9.99201e-16 -6.64746e-15  
-2.22045e-15  1.0      4.7462e-15  
-2.55351e-15 -8.88178e-16  1.0
```

Матрица 1000 x 1000:

n = 1000

A = randn(n,n)

1000x1000 Matrix{Float64}:

```
-0.14442 -1.36559 -1.11578  ...  0.437918 -1.20718 -0.2666  
-1.85655 -0.453664 -0.0178234 -0.295606 -0.661408 0.0013573  
0.356631 0.959434 0.427806 1.03361 -0.0627254 -0.460862  
1.07163 0.921197 0.648523 2.23529 1.34531 0.987827  
0.947356 1.13044 -0.3468 0.974637 0.555962 0.750848  
1.2813 0.259945 1.56598 -0.522589 0.0692786 -1.52923  
0.187527 0.0855509 1.75435 -0.992476 0.388785 -0.12862  
0.201255 0.701648 -1.05067 -0.573835 0.0284804 0.340921  
0.47958 -0.342495 0.786705 0.590131 -0.751631 -1.22405  
-0.854317 -2.19158 -1.11672 1.1334 -0.534909 2.28377  
-0.438878 -1.54925 0.881022 1.05579 0.118616 -0.857989  
-0.168349 0.0480901 0.0133083 -0.823289 -0.0602442 0.565706  
-1.36284 0.359569 -0.324999 -0.900429 -0.801126 0.269208
```

Выполнение лабораторной работы

```
import Pkg
Pkg.add("BenchmarkTools")
using BenchmarkTools

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

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

128.290 ms (21 allocations: 7.99 MiB)

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

771.415 ms (27 allocations: 7.93 MiB)

```
# Оценка эффективности выполнения операции по нахождению
# собственных значений зашумлённой матрицы,
# для которой явно указано, что она симметричная:
@btime eigvals(Asym_explicit);
```

129.169 ms (21 allocations: 7.99 MiB)

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

```
1000000x1000000 SymTridiagonal{Float64, Vector{Float64}}:
 1.25177 -1.95644      .      .      .      .      .
-1.95644 -0.857361  0.33407      .      .      .      .
      .  0.33407   2.25289 -1.08965      .      .      .
      .      . -1.08965 -0.24222      .      .      .
      .      .      .  2.00241      .      .      .
      .      .      .      .      .      .      .
      .      .      .      .      .      .      .
      .      .      .      .      .      .      .
      .      .      .      .      .      .      .
```

Выполнение лабораторной работы

```
{ |  
    # Матрица с рациональными элементами:  
    Arational = Matrix{Rational{BigInt}}(rand(1:10, 3, 3))/10  
  
    3x3 Matrix{Rational{BigInt}}:  
    1//2  4//5  1//2  
    9//10 9//10 1//2  
    3//5   1   7//10  
  
    # Единичный вектор:  
    x = fill(1, 3)  
  
    3-element Vector{Int64}:  
    1  
    1  
    1  
  
    # Задаём вектор b:  
    b = Arational*x  
  
    3-element Vector{Rational{BigInt}}:  
    9//5  
    23//10  
    23//10  
  
    # Решение исходного уравнения получаем с помощью функции \  
    # (убеждаемся, что x - единичный вектор):  
    Arational\b  
  
    3-element Vector{Rational{BigInt}}:  
    1  
    1  
    1  
  
    # LU-разложение:  
    lu(Arational)  
  
    LU{Rational{BigInt}, Matrix{Rational{BigInt}}, Vector{Int64}}  
    L factor:  
    3x3 Matrix{Rational{BigInt}}:  
    1  0  0  
    1//2  0  0  
    1//2  0  0
```

```
# 1. Задаем вектор v
v = [1, 2, 3, 4, 5] # можно использовать любой вектор

# Скалярно умножаем вектор v сам на себя и сохраняем результат в dot_v
dot_v = dot(v, v)

55
```

Рис. 9: Произведение векторов

```
# 2.  
outer_v = v * v' # используем библ LinearAlgebra
```

```
5x5 Matrix{Int64}:
```

```
1  2  3  4  5  
2  4  6  8 10  
3  6  9 12 15  
4  8 12 16 20  
5 10 15 20 25
```

Рис. 10: Произведение векторов

```
# Система:  
#  $x + y = 2$   
#  $x - y = 3$   
A1 = [1 1; 1 -1]
```

```
Alu1 = lu(A1)
```

```
b1 = [2, 3]
```

```
slaul = A1\b1  
show(slaul)
```

```
Alu1\b1
```

```
→ [2.5, -0.5]  
2-element Vector{Float64}:  
 2.5  
-0.5
```

Выполнение лабораторной работы

```
# Система:  
#  $x + y = 2$   
#  $2x + 2y = 4$   
  
# Матрица коэффициентов A  
A2 = [1 1;  
      2 2]  
  
# Вектор правых частей b  
b2 = [2, 4]  
  
if (det(A2) != 0)  
    slau2 = A2\b2  
    print(slau2)  
else  
    print("Нет решений")  
end
```

Нет решений

```
# c  
A3 = [1 1; 2 2]  
b3 = [2, 5]  
if (det(A3) != 0)  
    slau3 = A3\b3  
    print(slau3)  
else  
    print("Нет решений")  
end
```

Нет решений

```
# d  
A4 = [1 1; 2 2; 3 3]  
b4 = [1, 2, 3]  
slau4 = A4\b4
```

```
2-element Vector{Float64}:  
 0.4999999999999999
```

Выполнение лабораторной работы

```
# d
A4 = [1 1; 2 2; 3 3]
b4 = [1, 2, 3]
slau4 = A4\b4
```

```
2-element Vector{Float64}:
 0.4999999999999999
 0.5
```

```
# e
A1 = [1 1; 2 1; 1 -1]
b1 = [2, 1, 3]
slau1 = A1\b1
```

```
2-element Vector{Float64}:
 1.5000000000000004
-0.9999999999999997
```

```
# f
A2 = [1 1; 2 1; 3 2]
b2 = [2, 1, 3]
slau2 = A2\b2
```

```
2-element Vector{Float64}:
-0.9999999999999989
 2.999999999999982
```


Выполнение лабораторной работы

```
# a
A3 = [1 1 1; 1 -1 -2]
b3 = [2, 3]
slau2 = A3\b3
```

```
3-element Vector{Float64}:
 2.2142857142857144
 0.35714285714285704
-0.5714285714285712
```

```
# b
A2 = [1 1 1; 2 2 -3; 3 1 1]
b2 = [2, 4, 1]
slau2 = A2\b2
```

```
3-element Vector{Float64}:
-0.5
 2.5
 0.0
```

```
# c
A3 = [1 1 1; 1 1 2; 2 2 3]
b3 = [1, 0, 1]
if (det(A3) != 0)
    slau3 = A3\b3
    print(slau3)
else
    print("Нет решений")
end
```

Нет решений

```
# d
A3 = [1 1 1; 1 1 2; 2 2 3]
b3 = [1, 0, 0]
if (det(A3) != 0)
    slau3 = A3\b3
    print(slau3)
else
    print("Нет решений")
end
```

Нет решений

Выполнение лабораторной работы

```
A = [1 -2; -2 1]
AsymEig = eigen(A)
display(diagm(AsymEig.values))
```

```
2x2 Matrix{Float64}:
-1.0  0.0
 0.0  3.0
```

```
B = [1 -2; -2 3]
Beig = eigen(B)
display(diagm(Beig.values))
```

```
2x2 Matrix{Float64}:
-0.236068  0.0
 0.0       4.23607
```

```
C = [1 -2 0; -2 1 2; 0 2 0]
Ceig = eigen(C)
display(diagm(Ceig.values))
```

```
3x3 Matrix{Float64}:
-2.14134  0.0  0.0
 0.0      0.515138  0.0
 0.0      0.0  3.6262
```

Выполнение лабораторной работы

```
A = [1 -2; -2 1]
```

```
A^10
```

```
2x2 Matrix{Int64}:
```

```
29525  -29524  
-29524  29525
```

```
B = [5 -2; -2 5]
```

```
sqrt_B = sqrt(B)
```

```
2x2 Matrix{Float64}:
```

```
2.1889  -0.45685  
-0.45685  2.1889
```

```
sqrt_B * sqrt_B
```

```
2x2 Matrix{Float64}:
```

```
5.0  -2.0  
-2.0  5.0
```

```
C = [1 -2; -2 1]
```

```
C^(1/3)
```

```
2x2 Symmetric{ComplexF64, Matrix{ComplexF64}}:
```

```
0.971125+0.433013im  -0.471125+0.433013im  
-0.471125+0.433013im  0.971125+0.433013im
```

```
D = [1 2; 2 3]
```

```
sqrt(D)
```

```
2x2 Matrix{ComplexF64}:
```

```
0.568864+0.351578im  0.920442-0.217287im  
0.920442-0.217287im  1.48931+0.134291im
```

Выполнение лабораторной работы

```
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]
```

```
5x5 Matrix{Int64}:
```

```
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
```

```
Aeig = eigen(A)
diagm(Aeig.values)
```

```
5x5 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
```

```
LowerTriangular(A)
```

```
5x5 LowerTriangular{Int64, Matrix{Int64}}:
```

```
140  .  .  .  .
 97 106  .  .  .
 74  89 152  .  .
168 131 144  54  .
131  36  71 142 36
```

Рис. 17: Операции с матрицами

Выполнение лабораторной работы

```
a = [1 2; 3 4]
b = 1/2*a
c = 1/10*a
E = Matrix(I, 2, 2)
```

```
2x2 Matrix{Bool}:
 1  0
 0  1
```

```
inv(E - a) # Не продуктивная
```

```
2x2 Matrix{Float64}:
-0.0  -0.333333
-0.5   0.0
```

```
inv(E - b) # Не продуктивная
```

```
2x2 Matrix{Float64}:
-0.4  -0.8
-1.2  -0.4
```

```
inv(E - c) # продуктивная
```

```
2x2 Matrix{Float64}:
 1.2  0.266667
 0.4  1.2
```

Выполнение лабораторной работы

```
a = [1 2; 3 1]
b = 1/2*a
c = 1/10*a
E = Matrix(I, 2, 2)
```

```
2x2 Matrix{Bool}:
 1  0
 0  1
```

```
inv(E - a)^-1 # Не продуктивная
```

```
2x2 Matrix{Float64}:
-0.0  -2.0
-3.0   0.0
```

```
inv(E - b)^-1 # Не продуктивная
```

```
2x2 Matrix{Float64}:
 0.5  -1.0
-1.5   0.5
```

```
inv(E - c)^-1 # Не продуктивная
```

```
2x2 Matrix{Float64}:
 0.9  -0.2
-0.3   0.9
```

Выполнение лабораторной работы

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

```
3x3 Matrix{Float64}:
```

```
0.1  0.2  0.3  
0.0  0.1  0.2  
0.0  0.1  0.3
```

```
aeig = eigen(a)  
abs.(aeig.values) < 1
```

```
2-element BitVector:
```

```
0  
0
```

```
beig = eigen(b)  
abs.(beig.values) < 1
```

```
2-element BitVector:
```

```
1  
0
```

```
ceig = eigen(c)  
abs.(ceig.values) < 1
```

```
2-element BitVector:
```

```
1  
1
```

```
deig = eigen(d)  
abs.(deig.values) < 1
```

```
3-element BitVector:
```

```
1  
1  
1
```

В процессе выполнения данной лабораторной работы я изучила возможности специализированных пакетов Julia для выполнения и оценки эффективности операций над объектами линейной алгебры.