

# Компьютерный практикум по статистическому анализу данных

Лабораторная работа № 4. Линейная алгебра

Демидова Екатерина Алексеевна

# Содержание

1	Введение	4
2	Теоретическое введение	5
3	Выполнение лабораторной работы	6
4	Выводы	14
	Список литературы	15

## Список иллюстраций

3.1	Поэлементные операции над многомерными массивами . . . . .	6
3.2	Примеры. Транспонирование, след, ранг, определитель и инверсия матрицы . . . . .	7
3.3	Примеры. Вычисление нормы векторов и матриц, повороты, вращения . . . . .	7
3.4	Примеры. Матричное умножение, единичная матрица, скалярное произведение . . . . .	8
3.5	Примеры. Факторизация. Специальные матричные структуры . .	8
3.6	Примеры. Факторизация. Специальные матричные структуры . .	9
3.7	Примеры. Общая линейная алгебра . . . . .	9
3.8	Задание 1, 2 . . . . .	10
3.9	Задание 2 . . . . .	10
3.10	Задание 3 . . . . .	11
3.11	Задания 3 . . . . .	11
3.12	Задание 3 . . . . .	12
3.13	Задания 4 . . . . .	12
3.14	Задание 4 . . . . .	13

# 1 Введение

## Цель работы

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

## Задачи

1. Используя Jupyter Lab, повторите примеры.
2. Выполните задания для самостоятельной работы.

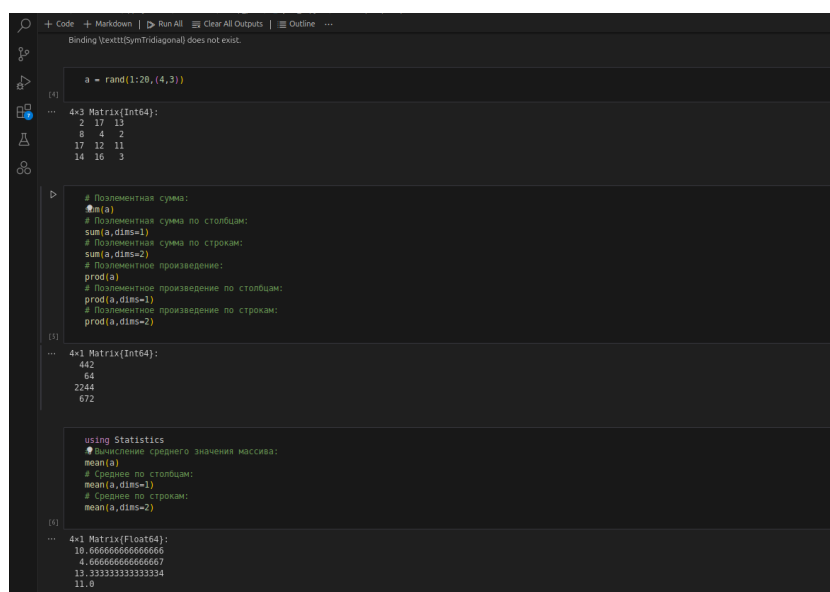
## 2 Теоретическое введение

Julia — высокоуровневый свободный язык программирования с динамической типизацией, созданный для математических вычислений.[1]. Эффективен также и для написания программ общего назначения. Синтаксис языка схож с синтаксисом других математических языков, однако имеет некоторые существенные отличия.

Для выполнения заданий была использована официальная документация Julia[2].

### 3 Выполнение лабораторной работы

Выполним примеры из лабораторной работы для изучения циклов и функций(рис. 3.1 - 3.7)



```
+ Code + Markdown | ▶ Run All | Clear All Outputs | Outline ...
Binding \texttt{SymTridiagonal} does not exist.

a = rand(1:20,(4,3))

[1]
... 4x3 Matrix[Int64]:
      2  17  13
      8   4   2
     17  12  11
     14  16   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)

[1]
... 4x1 Matrix[Int64]:
    442
     64
    2244
     672

using Statistics
# Вычисление среднего значения массива:
mean(a)
# Среднее по столбцам:
mean(a,dims=1)
# Среднее по строкам:
mean(a,dims=2)

[1]
... 4x1 Matrix[Float64]:
 10.666666666666666
  4.666666666666667
 13.333333333333334
 11.0
```

Рис. 3.1: Поэлементные операции над многомерными массивами

```

using LinearAlgebra

# Массив 4x4 со случайными целыми числами (от 1 до 20):
a = rand(1:20,(4,4))

# Транспонирование:
transpose(b)
# След матрицы (сумма диагональных элементов):
tr(b)
# Разложение диагональных элементов как массива:
diag(b)
# Ранг матрицы:
rank(b)
# Инверсия матрицы (определение обратной матрицы):
inv(b)
# Определитель матрицы:
det(b)
# Псевдообратная функция для прямоугольных матриц:
pinv(a)

3x4 Matrix{Float64}:
-0.0451278  0.0252084  0.0470064  0.00639139
 0.0342491 -0.0126234 -0.0580323  0.0727919
 0.0378354  0.00118211  0.0725937 -0.0940724

# Создание вектора X:
x = [2, 4, -5]
# Вычисление евклидовой нормы:
norm(x)
# Вычисление p-нормы:
p = 1
norm(x,p)
# Расстояние между двумя векторами X и Y:
X = [2, 4, -5];
Y = [1,-1,3];
norm(X-Y)
# Проверка по базовому определению:
sqrt(sum((X-Y).^2))

9.486832989595138

```

Рис. 3.2: Примеры. Транспонирование, след, ранг, определитель и инверсия матрицы

```

# Угол между двумя векторами:
acos((transpose(X)*Y)/(norm(X)*norm(Y)))

# Создание матрицы:
d = [5 -4 2 ; -1 2 3; -2 1 0]
# Вычисление Евклидовой нормы:
norm(d)
# Вычисление p-нормы:
p=1
norm(d,p)

8.0

# Поворот на 180 градусов:
rot180(d)
# Переворачивание строк:
reverse(d,dims=1)
# Переворачивание столбцов:
reverse(d,dims=2)

3x3 Matrix{Int64}:
 2  -4  5
 3  2 -1
 0  1 -2

# Матрица 3x3 со случайными целыми значениями от 1 до 10:
A = rand(1:10,(3,3))
# 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

-17

```

Рис. 3.3: Примеры. Вычисление нормы векторов и матриц, повороты, вращения

```
+ Code + Markdown | ▶ Run All | Clear All Outputs | Outline ...

A = rand(3, 3)
b = fill(1.0, 3)
b = A*b

[13]
3-element Vector{Float64}:
 1.2402436626965459
 1.3590231766385592
 1.9663382828283189

A\b

[14]
3-element Vector{Float64}:
 1.0000000000000002
 0.9999999999999974
 1.0000000000000038

Alu = lu(A)

[15]
LU{Float64, Matrix{Float64}, Vector{Int64}}
L factor:
3x3 Matrix{Float64}:
 1.0  0.0  0.0
 0.132116 1.0  0.0
 0.282647 0.605518 1.0
U factor:
3x3 Matrix{Float64}:
 6.92016 0.911462 0.124709
 0.0  0.659235 0.44095
 0.0  0.0  0.018534

Alu\b

[16]
3-element Vector{Float64}:
 1.0000000000000002
 0.9999999999999974
 1.0000000000000038

det(Alu)

[17]
-0.011544281453697876
```

Рис. 3.4: Примеры. Матричное умножение, единичная матрица, скалярное произведение

```
+ Code + Markdown | ▶ Run All | Clear All Outputs | Outline ...

inv(AsymEig)*Asym

[21]
3x3 Matrix{Float64}:
 1.0  5.5112e-15  -6.38378e-15
-1.44329e-15  1.0  9.29812e-16
-1.27676e-15 -1.44329e-15  1.0

n = 1000
A = randn(n,n)
Asym = A + A'
isymmetric(Asym)

[22]
true

Asym_noisy = copy(Asym)
Asym_noisy[1,2] += 5eps()
isymmetric(Asym_noisy)

[23]
false

Asym_explicit = Symmetric(Asym_noisy)

[24]
1000x1000 Symmetric{Float64, Matrix{Float64}}:
-1.18785  2.36754  1.7734  -0.888384 -2.67477  0.0710199
 2.36754  2.98918 -1.7326  -0.412284  0.927346  0.943464
 1.7734  -1.7326 -1.93827  0.0117656  0.362245 -1.26935
 0.0715804  0.321184  0.917852 -1.52921 -1.08014  0.880223
-1.44329e-15  1.0  9.29812e-16  2.31923  0.651896  0.384824
-1.27676e-15 -1.44329e-15  1.0 -0.438759 -1.26461  0.499111
-1.18698 -1.06752 -1.06744 -0.438759 -1.26461  0.499111
 0.966376  1.32647 -0.142847 -1.01087 -0.169422  1.74265
-0.0173167 -0.623335  0.0780364  1.15388  0.0153348 -2.93588
-0.0612332  0.119189  0.409365  1.71291  1.40875  2.19677
 0.52526 -0.918453 -3.16274 -0.830246 -0.0907788  1.4746
 0.63662  1.13783 -2.21285 -1.06969  1.4799 -2.23543
 1.00422 -0.190182 -1.34838 -0.00426986  0.981785  1.63343
-0.307627 -2.31148 -1.65298  0.52584  2.93294 -2.22985
-0.381884  0.518722  2.2419  1.6852  0.351898 -0.0945664
-0.524858  1.25989 -0.744196 -2.41559  1.05148  2.46367
 0.593983 -1.07409 -0.453119 -0.715179  0.412781 -0.7988
 0.304591 -3.28029  0.119616  1.96255 -0.742149  0.652269
 2.411  2.40037  2.41776  0.888677  0.578399 -1.01182
-1.5141 -0.0188818  0.320218  1.97246 -0.318996 -1.605
-1.11468 -0.248895  1.29345 -2.2595  0.346618  0.456886
 2.37698 -1.79427 -0.39733 -2.86856  0.855489  0.61363
-0.676839  0.0158106 -0.097344  1.64082  1.47958 -1.01649
```

Рис. 3.5: Примеры. Факторизация. Специальные матричные структуры





### Задания для самостоятельного выполнения

```

v = [1, 2, 3]
dot_v = v*v
outer_v = v*v'

```

3x3 Matrix (Int64):

```

1 2 3
2 4 6
3 6 9

```

#### 4.4.2

##### № 1

```

A = [1 1; 1 -1]
B = [2; 3]
if det(A) == 0
    println("Решение не существует")
else
    println(A\B)
end

A = [1 1; 2 2]
B = [2; 4]
if det(A) == 0
    println("Решение не существует")
else
    println(A\B)
end

A = [1 1; 2 2]
B = [2; 3]
if det(A) == 0
    println("Решение не существует")
else
    println(A\B)
end

```

[2.5, -0.5]  
Решение не существует  
Решение не существует

Рис. 3.8: Задание 1, 2

```

A = [1 1; 2 2; 3 3]
B = [1; 2; 3]
println(A\B)

A = [1 1; 2 1; 1 -1]
B = [2; 1; 3]
println(A\B)

A = [1 1; 2 1; 3 2]
B = [2; 1; 3]
println(A\B)

```

[0.5, 0.5]  
[1.5000000000000004, -0.9999999999999997]  
[-0.9999999999999994, 2.9999999999999999]

##### № 2

```

A = [1 1 1; 1 -1 -2]
B = [2; 3]
println(A\B)

A = [1 1 1; 2 2 -3; 3 1 1]
B = [2; 4; 1]
println(A\B)

A = [1 1 1; 1 1 2; 2 2 3]
B = [1; 0; 1]
if det(A) == 0
    println("Решение не существует")
else
    println(A\B)
end

A = [1 1 1; 1 1 2; 2 2 3]
B = [1; 0; 0]
if det(A) == 0
    println("Решение не существует")
else
    println(A\B)
end

```

[2.214285714285715, 0.35714285714285715, -0.5714285714285711]  
[-0.5, 2.5, 0.0]  
Решение не существует  
Решение не существует

Рис. 3.9: Задание 2

```

№ 1

A = [1 -2; -2 1]
#eig = eigen(A)
A_diag = inv(Aeig.vectors)*A*Aeig.vectors
A_diag = diagm(Aeig.values)
display(A_diag)
A = [1 -2; -2 3]
Aeig = eigen(A)
A_diag = diagm(Aeig.values)
display(A_diag)
A = [1 -2 0; -2 1 2; 0 2 0]
Aeig = eigen(A)
A_diag = diagm(Aeig.values)
display(A_diag)

[40]
... 2x2 Matrix{Float64}:
-1.0  0.0
 0.0  3.0

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

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

№ 2

D
A = [1 -2; -2 1]
#display(A^10)
A = [5 -2; -2 5]
display(sqrt(A))
A = [1 -2; -2 1]
display(A^(1/3))
A = [1 2; 2 3]
display(sqrt(A))

[41]
... 2x2 Matrix{Int64}:
29525 -29524
-29524 29525

... 2x2 Matrix{Float64}:
2.1889 -0.45685
-0.45685 2.1889

... 2x2 Symmetric{ComplexF64, Matrix{ComplexF64}}:
0.071125e0 0.433013im -0.471125e0 0.433013im
-0.471125e0 0.433013im 0.071125e0 0.433013im

```

Рис. 3.10: Задание 3

```

№ 3

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]

[42]
... 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)

[46]
... Eigen{Float64, Float64, Matrix{Float64}, Vector{Float64}}
values:
5-element Vector{Float64}:
-128.49322764882147
-55.88778455385702
42.752167279318826
87.16111477314497
542.4677301466136
vectors:
5x5 Matrix{Float64}:
-0.147575  0.647178  0.018882  0.548903 -0.587987
-0.256795 -0.173068  0.834628 -0.239864 -0.387253
-0.185537  0.229762 -0.422161 -0.731925 -0.448631
0.810794 -0.247586 -0.0273194  0.8366447 -0.514526
-0.453805 -0.657619 -0.352577  0.322668 -0.364928

D
diagm(Aeig.values)

[49]
... 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)

[52]
... 5x5 LowerTriangular{Int64, Matrix{Int64}}:
140 . . . .
97 106 . . .
74 89 152 . .
168 131 144 54

```

Рис. 3.11: Задания 3

```

@btime diagm(Aeig.values)
[54]
... 109.921 ns (1 allocation: 256 bytes)

... 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

@btime LowerTriangular(A)
[63]
... 59.206 ns (1 allocation: 16 bytes)

... 5x5 LowerTriangular{Int64, Matrix{Int64}}:
140  -  -  -
 97 106  -  -
 74  89 152  -
168 131 144  54
131  36  71 142  36

```

Рис. 3.12: Задание 3

```

№1

$$x - Ax = y$$


A = [1 2; 3 4]
B = (1/2)*[1 2; 3 4]
C = (1/10)*[1 2; 3 4]
[98]
... 2x2 Matrix{Float64}:
0.1 0.2
0.3 0.4

№2

A = [1 2; 3 1]
B = (1/2)*[1 2; 3 1]
C = (1/10)*[1 2; 3 1]
E = Matrix{I,2,2}
[17]
... 2x2 Matrix{Bool}:
1 0
0 1

inv(E-A) #не продуктивная
[99]
... 2x2 Matrix{Float64}:
0.5 -0.333333
-0.5 0.0

inv(E-B) #не продуктивная
[100]
... 2x2 Matrix{Float64}:
0.5 -0.5
-0.75 -0.25

inv(E-C) #продуктивная
[101]
... 2x2 Matrix{Float64}:
1.25 0.416667
0.625 1.875

```

Рис. 3.13: Задания 4

```

№3

D = [0.1 0.2 0.3; 0 0.1 0.2; 0 0.1 0.3]

[93]
... 3x3 Matrix{Float64}:
 0.1 0.2 0.3
 0.0 0.1 0.2
 0.0 0.1 0.3

abs.(eigen(A).values).<1 #не продуктивная

[94]
... 2-element BitVector:
 1
 0

abs.(eigen(B).values).<1 #не продуктивная

[95]
... 2-element BitVector:
 1
 0

abs.(eigen(C).values).<1 #продуктивная

[96]
... 2-element BitVector:
 1
 1

abs.(eigen(D).values).<1 #продуктивная

[97]
... 3-element BitVector:
 1
 1
 1

D> ?eye

[98]
... search: KeyError eltype keytype supertype code_typed supertypes @code_typed
end
end
end

```

Рис. 3.14: Задание 4

## 4 Выводы

В результате выполнения работы освоили применение циклов функций и сторонних для Julia пакетов для решения задач линейной алгебры и работы с матрицами.

## Список литературы

1. JuliaLang [Электронный ресурс]. 2024 JuliaLang.org contributors. URL: <https://julialang.org/> (дата обращения: 11.10.2024).
2. Julia 1.11 Documentation [Электронный ресурс]. 2024 JuliaLang.org contributors. URL: <https://docs.julialang.org/en/v1/> (дата обращения: 11.10.2024).