Лабораторная работа №4

Линейная алгебра

Клюкин М. А.

Российский университет дружбы народов, Москва, Россия

Докладчик

- Клюкин Михаил Александрович
- студент
- Российский университет дружбы народов
- 1132226431@pruf.ru
- https://MaKYaro.github.io/ru/

Цель работы

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

Задание

1. Используя Jupyter Lab, повторитm примеры из раздела 4.2.
2. Выполнитm задания для самостоятельной работы (раздел 4.4).

Выполнение лабораторной работы

Поэлементные операции над многомерными массивами

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

```
4×3 Matrix{Int64}:
  1   7  17
 13  11  19
 18  15   6
 10  10  19
```

```
sum(a)
```

```
146
```

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

```
1×3 Matrix{Int64}:
 42  43  61
```

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

```
4×1 Matrix{Int64}:
 25
 43
 39
 39
```

Рис. 1: Пример поэлементного суммирования по столбцам и строкам

Поэлементные операции над многомерными массивами

```
prod(a)
```

```
995188194000
```

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

```
1×3 Matrix{Int64}:
 2340  11550  36822
```

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

```
4×1 Matrix{Int64}:
   119
  2717
  1620
  1900
```

Рис. 2: Пример поэлементного произведения по столбцам и строкам

```
import Pkg
Pkg.add("Statistics")
```

```
    Updating registry at `~/.julia/registries/General.toml`
   Resolving package versions...
    Updating `~/.julia/environments/v1.11/Project.toml`
  [10745b16] + Statistics v1.11.1
  No Changes to `~/.julia/environments/v1.11/Manifest.toml`
```

```
using Statistics
```

Рис. 3: Импорт пакета Statistics

Поэлементные операции над многомерными массивами

```
mean(a)
```

```
12.166666666666666
```

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

```
1×3 Matrix{Float64}:
 10.5  10.75  15.25
```

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

```
4×1 Matrix{Float64}:
  8.333333333333334
 14.333333333333334
 13.0
 13.0
```

Рис. 4: Пример нахождения средних во всей матрице, по столбцам и по строкам

```
import Pkg
Pkg.add("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`
```

Рис. 5: Импорт пакета LinearAlgebra

Поэлементные операции над многомерными массивами

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

```
4×4 Matrix{Int64}:
 12  1  15  11
  5  1   1   4
 13  9   9  12
  7  4   7   5
```

```
transpose(b)
```

```
4×4 transpose(::Matrix{Int64}) with eltype Int64:
 12  5  13  7
  1  1   9  4
 15  1   9  7
 11  4  12  5
```

```
tr(b)
```

```
27
```

```
diag(b)
```

```
4-element Vector{Int64}:
 12
  1
  9
  5
```

Рис. 6: Транспонирование, след и диагональные элементы

Поэлементные операции над многомерными массивами

```
rank(b)
```

```
4
```

```
inv(b)
```

```
4×4 Matrix{Float64}:
 -0.0989011   0.416209  -0.240385    0.461538
 -0.0879121  -0.171703   0.105769    0.0769231
  0.0549451  -0.18956   -0.0192308   0.0769231
  0.131868   -0.179945   0.278846   -0.615385
```

```
det(b)
```

```
-728.0000000000006
```

```
pinv(a)
```

```
3×4 Matrix{Float64}:
 -0.149505    0.10118   -0.0331293   0.0430493
  0.183742   -0.138281   0.126337   -0.0660156
 -0.0102132   0.044137  -0.0498511   0.0333752
```

Рис. 7: Ранг, обратная матрица, детерминант и псевдообратная матрица

Вычисление нормы векторов и матриц, повороты, вращения

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

```
3-element Vector{Int64}:
  2
  4
 -5
```

```
norm(X)
```

```
6.708203932499369
```

```
p = 1
```

```
1
```

```
norm(X, p)
```

```
11.0
```

Рис. 8: Евклидова норма и p-норма вектора

Вычисление нормы векторов и матриц, повороты, вращения

```
X = [2, 4, -5]
Y = [1, -1, 3]
```

```
3-element Vector{Int64}:
  1
 -1
  3
```

```
norm(X-Y)
```

```
9.486832980505138
```

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

```
9.486832980505138
```

Рис. 9: Расстояние между векторами

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

```
2.4404307889469252
```

Рис. 10: Угол между векторами

Вычисление нормы векторов и матриц, повороты, вращения

```
d = [5 -4 2; -1 2 3; -2 1 0]
```

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

```
opnorm(d)
```

```
7.147682841795258
```

```
p = 1
opnorm(d, p)
```

```
8.0
```

```
rot180(d)
```

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

```
reverse(d, dims=1)
```

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

```
reverse(d, dims=2)
```

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

Рис. 11: Угол между векторами

Матричное умножение, единичная матрица, скалярное произведение

```
A = rand(1:10, (2,3))
B = rand(1:10, (3,4))

A*B
```

```
2×4 Matrix{Int64}:
 57  124  97  121
 50   54  76   72
```

```
Matrix{Int}(I, 3, 3)
```

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

```
X = [2,4,-5]
Y = [1,-1,3]
dot(X,Y)
```

```
-17
```

```
X'Y
```

```
-17
```

Рис. 12: Произведение матриц и скалярного векторов

Факторизация. Специальные матричные структуры

```
A = rand(3, 3)
```

```
3×3 Matrix{Float64}:
 0.761309   0.235826  0.107526
 0.0319393  0.444271  0.559785
 0.986689   0.117963  0.835225
```

```
x = fill(1.0, 3)
```

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

```
b = A*x
```

```
3-element Vector{Float64}:
 1.1046609034825534
 1.035994844574446
 1.9398770738702553
```

```
A\b
```

```
3-element Vector{Float64}:
 1.0
 0.9999999999999998
 1.0000000000000002
```

Рис. 13: Решение СЛАУ

Факторизация. Специальные матричные структуры

```
Alu = lu(A)
```

```
LU{Float64, Matrix{Float64}, Vector{Int64}}
L factor:
3×3 Matrix{Float64}:
 1.0        0.0       0.0
 0.0323702  1.0       0.0
 0.77158    0.328773  1.0
U factor:
3×3 Matrix{Float64}:
 0.986689  0.117963   0.835225
 0.0       0.440452   0.532749
 0.0       0.0       -0.712071
```

Рис. 14: Пример LU-факторизации

Факторизация. Специальные матричные структуры

```
Alu.P
```

```
3×3 Matrix{Float64}:
 0.0  0.0  1.0
 0.0  1.0  0.0
 1.0  0.0  0.0
```

```
Alu.p
```

```
3-element Vector{Int64}:
 3
 2
 1
```

```
Alu.L
```

```
3×3 Matrix{Float64}:
 1.0        0.0       0.0
 0.0323702  1.0       0.0
 0.77158    0.328773  1.0
```

```
Alu.U
```

```
3×3 Matrix{Float64}:
 0.986689  0.117963   0.835225
 0.0       0.440452   0.532749
 0.0       0.0       -0.712071
```

Рис. 15: Пример извлечения различных частей факторизации

```
A\b
```

```
3-element Vector{Float64}:
 1.0
 0.9999999999999998
 1.0000000000000002
```

```
Alu\b
```

```
3-element Vector{Float64}:
 1.0
 0.9999999999999998
 1.0000000000000002
```

Рис. 16: Пример решения системы с использованием объектов факторизации

```
det(A)
```

```
0.30945811236985504
```

```
det(Alu)
```

```
0.30945811236985504
```

Рис. 17: Пример нахождения детерминанта матрицы A

```
Aqr = qr(A)
```

```
LinearAlgebra.QRCompactWY{Float64, Matrix{Float64}, Matrix{Float64}}
Q factor: 3×3 LinearAlgebra.QRCompactWYQ{Float64, Matrix{Float64}, Matrix{Float64}}
R factor:
3×3 Matrix{Float64}:
 -1.24666  -0.24876   -0.741056
  0.0      -0.452796  -0.415715
  0.0       0.0        0.548214
```

Рис. 18: Пример вычисления QR-факторизации

Факторизация. Специальные матричные структуры

```
Aqr.Q
```

```
3×3 LinearAlgebra.QRCompactWYQ{Float64, Matrix{Float64}, Matrix{Float64}}
```

```
Aqr.R
```

```
3×3 Matrix{Float64}:
 -1.24666  -0.24876   -0.741056
  0.0      -0.452796  -0.415715
  0.0       0.0        0.548214
```

Рис. 19: Пример извлечения различных частей QR-факторизации

Факторизация. Специальные матричные структуры

```
Aqr.Q'*Aqr.Q
```

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

```
Asym = A + A'
```

```
3×3 Matrix{Float64}:
 1.52262   0.267766  1.09421
 0.267766  0.888541  0.677748
 1.09421   0.677748  1.67045
```

Рис. 20: Проверка ортогональности и симметризация

Факторизация. Специальные матричные структуры

```
AsymEig = eigen(Asym)
```

```
Eigen{Float64, Float64, Matrix{Float64}, Vector{Float64}}
values:
3-element Vector{Float64}:
 0.31309603380927775
 0.8459673326154773
 2.9225465145784986
vectors:
3×3 Matrix{Float64}:
 -0.481025   0.619186  -0.620664
 -0.565358  -0.760163  -0.320191
  0.670064  -0.196878  -0.715719
```

```
AsymEig.values
```

```
3-element Vector{Float64}:
 0.31309603380927775
 0.8459673326154773
 2.9225465145784986
```

```
AsymEig.vectors
```

```
3×3 Matrix{Float64}:
 -0.481025   0.619186  -0.620664
 -0.565358  -0.760163  -0.320191
  0.670064  -0.196878  -0.715719
```

Рис. 21: Спектральное разложение, вычисление собственных векторов и собственных значений

```
inv(AsymEig)*Asym
```

```
3×3 Matrix{Float64}:
 1.0           1.11022e-16  -8.88178e-16
 0.0           1.0          -2.22045e-16
 1.33227e-15  -2.22045e-16   1.0
```

Рис. 22: Проверка на единичную матрицу

Факторизация. Специальные матричные структуры

```
n = 1000
A = randn(n,n)
```

```
1000×1000 Matrix{Float64}:
  1.21919     1.89498     1.13769    …  -1.17839     0.329318    0.907566
 -0.0667729  -0.411892    0.623627       -0.497358    0.245039   -0.672191
  1.97938     0.259871    1.84552         3.37711    -0.821011    0.190007
 -0.107106   -1.41858     1.33891        -0.560825    2.34398    -0.793936
  1.11746    -1.40773     0.962896       -0.444347    0.365432   -0.246332
  1.11887     1.09792    -0.101207   …    0.350729    1.22148    -0.164163
 -0.073545    1.42571     0.154895       1.67735     2.45225    -0.0106538
  0.480368   -0.0140386   0.742063      -1.16037     0.994236   -0.866863
  0.760408   -1.03767    -1.73576       -0.0114045  -0.376214    1.5636
 -0.271676   -0.362593    0.576431      -0.022571   -1.64629    -0.00603395
  0.23883    -0.34432     0.25431    …   0.486761    1.26078    -0.877192
 -0.993476   -1.30491     0.594969       1.02095     0.0595188  -3.62627
 -1.10402    -1.91583     0.650678       1.87037     1.63159    -0.236448
  ⋮                                   ⋱
  1.4279     -0.750515    1.98692       -1.97432    -0.883463    0.49891
  1.30728     0.373269    0.128268      -0.74715     1.24526     0.0595394
 -0.348066    0.70988    -0.290324   …   1.97024     0.30851    -1.03126
  0.371834   -0.0222176   1.52004        2.64894    -2.0726     -1.18298
  0.855957   -0.337223   -0.341019       0.0941659   0.58388    -0.850305
 -2.08439    -1.3649     -0.272118       0.361011    0.824685    0.0195274
 -1.54441     2.38897     1.93206        1.08577    -1.61254    -0.492179
  0.17855    -0.25958     0.187685   …  -1.34254     0.569655   -0.807811
 -0.210189    1.54664     2.14117       -0.525221   -1.40982    -0.472773
  0.490477    0.803903    0.0604441     -1.85743     0.165817   -0.99406
  1.10786    -1.25147    -0.943444       0.490933   -0.795693   -1.06095
 -0.710635    0.265125   -1.47829       -0.818611    1.70061     0.306508
```

Рис. 23: Создание матрицы большой размерности

Факторизация. Специальные матричные структуры

```
Asym = A + A'
```

```
1000×1000 Matrix{Float64}:
  2.43838     1.82821     3.11708    …  -0.687918   1.43718     0.196931
  1.82821    -0.823784    0.883498       0.306545  -1.00644    -0.407066
  3.11708     0.883498    3.69103        3.43756   -1.76445    -1.28829
  0.598679   -0.963684    1.50988       -0.430181   2.84789    -0.859776
  1.17454    -0.998664   -1.12439        0.464289  -0.459551    1.23509
  2.35132     2.46954     0.332644   …   0.712383   0.33854    -0.490996
 -1.35709     0.942664   -0.420858       1.0801     1.03252    -0.0416528
  0.367668   -0.0363813   1.60899       -1.53839    0.505019   -0.753032
 -0.0939787  -0.825324   -2.1775         0.910012  -1.38799     1.90886
 -0.240036   -0.183609    0.956247      -0.137951  -2.67164     0.1545
 -1.29718     0.895563   -1.07482    …  -1.52854    0.387326   -1.85323
 -1.34603    -0.0672108  -1.58795        0.9738    -0.606323   -3.10813
 -1.84127    -2.00403     0.422147      -0.130438   2.15705    -0.498119
  ⋮                                   ⋱
  0.29644    -0.556187    2.65114       -1.13435   -0.331933    0.212831
  1.4804      0.334041    1.1068         1.31372    1.36192    -0.445898
  0.797231    0.0531385  -0.0825638  …   2.29679    2.61058    -0.968592
 -0.0787694   0.0456341   1.56732        2.66166   -0.815377   -0.297262
  1.19047     0.559443    0.21221       -1.10113    2.90875     0.68196
 -0.494734   -2.9761     -1.01835        0.554991   1.29768     0.113033
 -3.35755     3.61254     3.14246        1.82621   -2.25072     1.17839
  0.142277    0.811109   -0.437482   …  -0.515118   1.35612    -0.694816
 -0.740635    1.21072     2.16774        1.21284   -1.37234    -2.08361
 -0.687918    0.306545    3.43756       -3.71485    0.65595    -1.81267
  1.43718    -1.00644    -1.76445        0.65595   -1.59139     0.639658
  0.196931   -0.407066   -1.28829       -1.81267    0.639658    0.613017
```

```
issymmetric(Asym)
```

Рис. 24: Симметризация матрицы A

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

```
1.8282068822135364
```

```
issymmetric(Asym_noisy)
```

```
false
```

Рис. 25: Пример добавления шума к симметричной матрице

Факторизация. Специальные матричные структуры

```
Asym_explicit = Symmetric(Asym_noisy)
```

```
1000×1000 Symmetric{Float64, Matrix{Float64}}:
  2.43838     1.82821     3.11708     …  -0.687918    1.43718     0.196931
  1.82821    -0.823784    0.883498        0.306545   -1.00644    -0.407066
  3.11708     0.883498    3.69103         3.43756    -1.76445    -1.28829
  0.598679   -0.963684    1.50988        -0.430181    2.84789    -0.859776
  1.17454    -0.998664   -1.12439         0.464289   -0.459551    1.23509
  2.35132     2.46954     0.332644    …   0.712383    0.33854    -0.490996
 -1.35709     0.942664   -0.420858        1.0801     1.03252    -0.0416528
  0.367668   -0.0363813   1.60899        -1.53839    0.505019   -0.753032
 -0.0939787  -0.825324   -2.1775          0.910012  -1.38799     1.90806
 -0.240036   -0.183609    0.956247       -0.137951  -2.67164     0.1545
 -1.29718     0.895563   -1.07482    …   -1.52854    0.387326   -1.85323
 -1.34603    -0.0672108  -1.58795         0.9738    -0.606323   -3.10813
 -1.84127    -2.00403     0.422147       -0.130438   2.15705    -0.498119
  ⋮                                   ⋱
  0.29644    -0.556187    2.65114        -1.13435    -0.331933   0.212831
  1.4804      0.334041    1.1068          1.31372     1.36192    -0.445898
  0.797231    0.0531385  -0.0825638   …   2.29679     2.61058    -0.968592
 -0.0787694   0.0456341   1.56732         2.66166    -0.815377   -0.297262
  1.19047     0.559443    0.21221        -1.10113     2.90875     0.68196
 -0.494734   -2.9761     -1.01835         0.554991    1.29768     0.113033
 -3.35755     3.61254     3.14246         1.82621    -2.25072     1.17839
  0.142277    0.811109   -0.437482    …  -0.515118    1.35612    -0.694816
 -0.748635    1.21072     2.16774         1.21284    -1.37234    -2.08361
 -0.687918    0.306545    3.43756        -3.71485     0.65595    -1.81267
  1.43718    -1.00644    -1.76445         0.65595    -1.59139     0.639658
  0.196931   -0.407066   -1.28829        -1.81267     0.639658    0.613017
```

```
issymmetric(Asym_explicit)
```

```
true
```

Рис. 26: Явное создание симметричной матрицы

Факторизация. Специальные матричные структуры

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

```
    Resolving package versions...
     Installed BenchmarkTools ─ v1.6.3
      Updating `~/.julia/environments/v1.11/Project.toml`
  [6e4b80f9] + BenchmarkTools v1.6.3
      Updating `~/.julia/environments/v1.11/Manifest.toml`
  [6e4b80f9] + BenchmarkTools v1.6.3
  [9abbd945] + Profile v1.11.0
Precompiling project...
    4292.1 ms  ✓ BenchmarkTools
  1 dependency successfully precompiled in 6 seconds. 468 already precompiled.
```

```
@btime eigvals(Asym);
```

```
  81.161 ms (21 allocations: 7.99 MiB)
```

```
@btime eigvals(Asym_noisy);
```

```
  794.744 ms (27 allocations: 7.93 MiB)
```

```
@btime eigvals(Asym_explicit);
```

```
  75.848 ms (21 allocations: 7.99 MiB)
```

```
n = 100000;
A = SymTridiagonal(randn(n), randn(n-1))

@btime eigmax(A)
```

```
  70.633 ms (44 allocations: 18.31 MiB)
5.902125972191184
```

```
B = Matrix(A)
```

```
OutOfMemoryError()
```

Рис. 27: Оценка эффективности выполнения операций над матрицами большой размерости

Общая линейная алгебра

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

```
3×3 Matrix{Rational{BigInt}}:
 7//10  9//10  1//2
 9//10   1     2//5
 2//5   3//5   2//5
```

```
x = fill(1, 3)
```

```
3-element Vector{Int64}:
 1
 1
 1
```

```
b = Arational*x
```

```
3-element Vector{Rational{BigInt}}:
 21//10
 23//10
  7//5
```

```
Arational\b
```

```
3-element Vector{Rational{BigInt}}:
 1
 1
 1
```

Рис. 28: Решение СЛАУ с рациональными элементами

```
lu(Arational)
```

```
LU{Rational{BigInt}, Matrix{Rational{BigInt}}, Vector{Int64}}
L factor:
3×3 Matrix{Rational{BigInt}}:
  1      0     0
 4//9    1     0
 7//9  11//14  1
U factor:
3×3 Matrix{Rational{BigInt}}:
 9//10   1     2//5
  0     7//45  2//9
  0      0     1//70
```

Рис. 29: LU разложение матрицы A

```
v = [1, 2, 33]
dot_v = dot(v, v)
```

```
1094
```

Рис. 30: Выполнение задания 1

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

```
3×3 Matrix{Int64}:
  1   2    33
  2   4    66
 33  66  1089
```

Рис. 31: Выполнение задания 1

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

```
3×3 Matrix{Int64}:
  1   2    33
  2   4    66
 33  66  1089
```

Рис. 32: Выполнение задания 1

```
A1 = [1 1; 1 -1]
b1 = [2; 3]
A1\b1
```

```
2-element Vector{Float64}:
  2.5
 -0.5
```

Рис. 33: Выполнение задания 2

```
A2 = [1 1; 2 2]
b2 = [2; 4]
det(A2)
```

```
0.0
```

Рис. 34: Выполнение задания 2

```
A3 = [1 1; 2 2]
b3 = [2 5]
det(A3)
```

```
0.0
```

Рис. 35: Выполнение задания 2

```
A4 = [1 1; 2 2; 3 3]
b4 = [1; 2; 3]
A4\b4
```

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

Рис. 36: Выполнение задания 2

```
A5 = [1 1; 2 1; 1 -1]
b5 = [2; 1; 3]
A5\b5
```

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

Рис. 37: Выполнение задания 2

```
A6 = [1 1; 2 1; 3 2]
b6 = [2; 1; 3]
A6\b6
```

```
2-element Vector{Float64}:
 -0.9999999999999989
  2.9999999999999982
```

Рис. 38: Выполнение задания 2

```
A1 = [1 1 1; 1 -1 -2]
b1 = [2; 3]
A1\b1
```

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

Рис. 39: Выполнение задания 2

Задания для самостоятельного выполнения

```
A2 = [1 1 1; 2 2 -3; 3 1 1]
b2 = [2; 4; 1]
A2\b2
```

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

Рис. 40: Выполнение задания 2

```
A3 = [1 1 1; 1 1 2; 2 2 3]
b3 = [1; 0; 1]
det(A3)
```

```
0.0
```

Рис. 41: Выполнение задания 2

```
A4 = [1 1 1; 1 1 2; 2 2 3]
b4 = [1; 0; 0]
det(A4)
```

```
0.0
```

Рис. 42: Выполнение задания 2

```
A = [1 -2; -2 1]
eigen_A = eigen(A)
diag_A = Diagonal(eigen_A.values)
```

```
2×2 Diagonal{Float64, Vector{Float64}}:
 -1.0    ⋅
   ⋅    3.0
```

Рис. 43: Выполнение задания 3

```
B = [1 -2; -2 3]
eigen_B = eigen(B)
diag_B = Diagonal(eigen_B.values)
```

```
2×2 Diagonal{Float64, Vector{Float64}}:
 -0.236068    ⋅
    ⋅        4.23607
```

Рис. 44: Выполнение задания 3

```
C = [1 -2 0; -2 1 0; 0 2 0]
eigen_C = eigen(C)
diag_C = Diagonal(eigen_C.values)
```

```
3×3 Diagonal{Float64, Vector{Float64}}:
 -1.0   .    .
   .   0.0   .
   .    .   3.0
```

Рис. 45: Выполнение задания 3

```
A = [1 -2; -2 1]
A ^ 10
```

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

Задания для самостоятельного выполнения

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

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

Рис. 47: Выполнение задания 3

Задания для самостоятельного выполнения

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

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

Рис. 48: Выполнение задания 3

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

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

Рис. 49: Выполнение задания 3

Задания для самостоятельного выполнения

```
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]

eigen_A = eigen(A)
```

```
Eigen{Float64, Float64, Matrix{Float64}, Vector{Float64}}
values:
5-element Vector{Float64}:
 -128.49322764802145
  -55.887784553057
   42.752167279318854
   87.16111477514488
  542.467730146614
vectors:
5×5 Matrix{Float64}:
 -0.147575   0.647178   0.010882    0.548903   -0.507907
 -0.256795  -0.173068   0.834628   -0.239864   -0.387253
 -0.185537   0.239762  -0.422161   -0.731925   -0.440631
  0.819704  -0.247506  -0.0273194   0.0366447  -0.514526
 -0.453805  -0.657619  -0.352577    0.322668   -0.364928
```

Задания для самостоятельного выполнения

```
diag_A = Diagonal(eigen_A.values)
```

```
5×5 Diagonal{Float64, Vector{Float64}}:
 -128.493      ⋅         ⋅         ⋅         ⋅
    ⋅      -55.8878     ⋅         ⋅         ⋅
    ⋅         ⋅       42.7522     ⋅         ⋅
    ⋅         ⋅         ⋅       87.1611     ⋅
    ⋅         ⋅         ⋅         ⋅       542.468
```

```
lt_A = LowerTriangular(A)
```

```
5×5 LowerTriangular{Int64, Matrix{Int64}}:
 140    ⋅    ⋅    ⋅   ⋅
  97  106    ⋅    ⋅   ⋅
  74   89  152    ⋅   ⋅
 168  131  144   54   ⋅
 131   36   71  142  36
```

```
@btime Diagonal(eigen_A.values)
@btime LowerTriangular(A)
```

```
  301.677 ns (1 allocation: 16 bytes)
  308.095 ns (1 allocation: 16 bytes)
5×5 LowerTriangular{Int64, Matrix{Int64}}:
 140    ⋅    ⋅    ⋅   ⋅
  97  106    ⋅    ⋅   ⋅
  74   89  152    ⋅   ⋅
 168  131  144   54   ⋅
 131   36   71  142  36
```

Рис. 51: Выполнение задания 3

```
A1 = [1 2; 3 4]
A2 = 1/2 * A1
A3 = 1/10 * A1
E = Matrix(I, 2, 2)
```

Рис. 52: Выполнение задания 4

Задания для самостоятельного выполнения

```
inv(E - A1)
```

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

```
inv(E - A2)
```

```
2×2 Matrix{Float64}:
  0.5   -0.5
 -0.75  -0.25
```

```
inv(E - A3)
```

```
2×2 Matrix{Float64}:
 1.25   0.416667
 0.625  1.875
```

Рис. 53: Выполнение задания 4

Задания для самостоятельного выполнения

```
A4 = [0.1 0.2 0.3;
0 0.1 0.2;
0 0.1 0.3]
```

```
3×3 Matrix{Float64}:
 0.1  0.2  0.3
 0.0  0.1  0.2
 0.0  0.1  0.3
```

```
abs.(eigen(A1).values).<1
```

```
2-element BitVector:
 1
 0
```

```
abs.(eigen(A2).values).<1
```

```
2-element BitVector:
 1
 0
```

```
abs.(eigen(A3).values).<1
```

```
2-element BitVector:
 1
 1
```

```
abs.(eigen(A4).values).<1
```

```
3-element BitVector:
 1
 1
 1
```

Рис. 54: Выполнение задания 4

Вывод

Изучили возможности специализированных пакетов Julia для выполнения и оценки эффективности операций над объектами линейной алгебры.