Отчёт по лабораторной работе №4

Компьютерный практикум по статистическому анализу данных

Канева Екатерина, НФИбд-02-22

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 Задание

- Используя Jupyter Lab, повторить примеры.
- Выполнить задания для самостоятельной работы.

3 Теоретическая часть

Julia - высокоуровневый свободный язык программирования с динамической типизацией, созданный для математических вычислений. 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.

4 Выполнение лабораторной работы

4.1 Примеры

Сначала я выполнила примеры с поэлементными операциями над многомерными массивами (рис. 4.1):

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

[67]: 4×3 Matrix{Int64}:
       16  18  14
       15   1   7
       10  11   7
        2  11  20

[69]: sum(a), sum(a,dims=1), sum(a,dims=2)

[69]: (132, [43 41 48], [48; 23; 28; 33;;])

[71]: prod(a), prod(a,dims=1), prod(a,dims=2)

[71]: (143434368000, [4800 2178 13720], [4032; 105; 770; 440;;])

[13]: import Pkg
      Pkg.add("Statistics")
      using Statistics

        Updating registry at `C:\Users\ekaneva\.julia\registries\General.toml`
       Resolving package versions...
        Updating `C:\Users\ekaneva\.julia\environments\v1.11\Project.toml`
      [10745b16] + Statistics v1.11.1
       No Changes to `C:\Users\ekaneva\.julia\environments\v1.11\Manifest.toml`

[73]: mean(a), mean(a,dims=1), mean(a,dims=2)

[73]: (11.0, [10.75 10.25 12.0], [16.0; 7.666666666666667; 9.333333333333334; 11.0;;])
```

Рис. 4.1: Первый раздел примеров.

Потом я выполнила примеры с транспонированием, следом, рангом, определителем и инверсией матрицы (рис. 4.2):

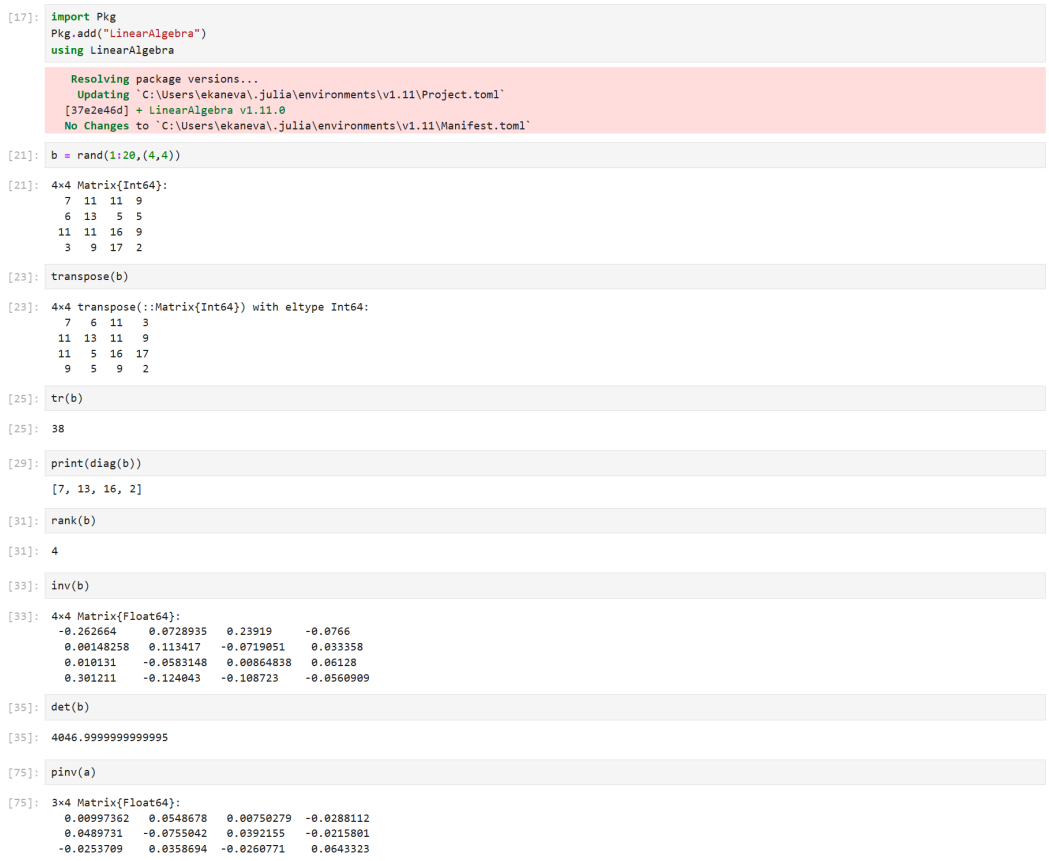

Рис. 4.2: Второй раздел примеров.

Потом я выполнила примеры с вычислением нормы векторов и матриц, поворотами, вращениями (рис. 4.3):

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

[43]: norm(X)

[43]: 6.708203932499369

[45]: p = 1
      norm(X,p)

[45]: 11.0

[49]: X = [2, 4, -5]
      Y = [1, -1, 3]
      norm(X - Y)

[49]: 9.486832980505138

[51]: sqrt(sum((X-Y).^2))

[51]: 9.486832980505138

[53]: acos((transpose(X)*Y)/(norm(X)*norm(Y)))

[53]: 2.4404307889469252

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

[57]: opnorm(d)

[57]: 7.147682841795258

[59]: p=1
      opnorm(d,p)

[59]: 8.0

[61]: rot180(d)

[61]: 3×3 Matrix{Int64}:
       0   1  -2
       3   2  -1
       2  -4   5

[63]: reverse(d,dims=1)

[63]: 3×3 Matrix{Int64}:
       -2   1  0
       -1   2  3
        5  -4  2
```

Рис. 4.3: Третий раздел примеров.

Потом я выполнила примеры с матричным умножением, единичной матрицей, скалярным произведением (рис. 4.4):

```
[77]: A = rand(1:10,(2,3))
      B = rand(1:10,(3,4));

[79]: A*B

[79]: 2×4 Matrix{Int64}:
       96  154  125  111
       96  107  122  100

[81]: Matrix{Int}(I, 3, 3)

[81]: 3×3 Matrix{Int64}:
       1  0  0
       0  1  0
       0  0  1

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

[83]: -17

[85]: X'Y

[85]: -17
```

Рис. 4.4: Четвёртый раздел примеров.

Потом я выполнила примеры с факторизацией, специальными матричными структурами (рис. 4.5-4.7):

```
[89]: A = rand(3, 3)
```

```
[89]: 3×3 Matrix{Float64}:
 0.702785   0.278485  0.722853
 0.32529    0.445412  0.399056
 0.0450734  0.104253  0.911617
```

```
[91]: x = fill(1.0, 3)
```

```
[91]: 3-element Vector{Float64}:
 1.0
 1.0
 1.0
```

```
[93]: b = A*x
```

```
[93]: 3-element Vector{Float64}:
 1.7041229810937706
 1.1697576850779718
 1.0609440466071827
```

```
[95]: A\b
```

```
[95]: 3-element Vector{Float64}:
 1.0000000000000002
 0.9999999999999994
 1.0000000000000002
```

```
[97]: Alu = lu(A)
```

```
[97]: LU{Float64, Matrix{Float64}, Vector{Int64}}
L factor:
3×3 Matrix{Float64}:
 1.0        0.0       0.0
 0.462858   1.0       0.0
 0.0641354  0.272951  1.0
U factor:
3×3 Matrix{Float64}:
 0.702785  0.278485  0.722853
 0.0       0.316513  0.0644779
 0.0       0.0       0.847658
```

```
[99]: Alu.p
```

```
[99]: 3-element Vector{Int64}:
 1
 2
 3
```

```
[101]: Alu\b
```

```
[101]: 3-element Vector{Float64}:
 1.0000000000000002
 0.9999999999999994
 1.0000000000000002
```

Рис. 4.5: Пятый раздел примеров (1).

```
[105]: det(A), det(Alu)

[105]: (0.18855350162195214, 0.18855350162195214)

[107]: Aqr = qr(A)

[107]: LinearAlgebra.QRCompactWY{Float64, Matrix{Float64}, Matrix{Float64}}
       Q factor: 3×3 LinearAlgebra.QRCompactWYQ{Float64, Matrix{Float64}, Matrix{Float64}}
       R factor:
       3×3 Matrix{Float64}:
        -0.775726  -0.445134  -0.875191
         0.0       -0.297776  -0.283805
         0.0        0.0        0.816274

[109]: Aqr.Q

[109]: 3×3 LinearAlgebra.QRCompactWYQ{Float64, Matrix{Float64}, Matrix{Float64}}

[111]: Asym = A + A'

[111]: 3×3 Matrix{Float64}:
        1.40557   0.603775  0.767926
        0.603775  0.890824  0.503309
        0.767926  0.503309  1.82323

[113]: AsymEig = eigen(Asym)

[113]: Eigen{Float64, Float64, Matrix{Float64}, Vector{Float64}}
       values:
       3-element Vector{Float64}:
        0.4918380908177505
        0.8956926855712033
        2.7320978776611917
       vectors:
       3×3 Matrix{Float64}:
         0.554209    -0.589624  -0.587534
        -0.832363    -0.396812  -0.386927
        -0.00499892   0.70348   -0.710698

[115]: inv(AsymEig)*Asym

[115]: 3×3 Matrix{Float64}:
         1.0          1.38778e-16   3.33067e-16
        -2.22045e-16  1.0          -1.66533e-16
         9.99201e-16  5.55112e-16   1.0
```

Рис. 4.6: Пятый раздел примеров (2).

```
[128]: n = 1000000;
       A = SymTridiagonal(randn(n), randn(n-1))

[128]: 1000000×1000000 SymTridiagonal{Float64, Vector{Float64}}:
        0.123032   0.802373   ⋅          ⋅         …   ⋅         ⋅          ⋅
        0.802373  -0.615898  2.57337     ⋅              ⋅         ⋅          ⋅
         ⋅         2.57337   0.61467    0.564902        ⋅         ⋅          ⋅
         ⋅          ⋅        0.564902  -1.28393         ⋅         ⋅          ⋅
         ⋅          ⋅         ⋅        -2.1384          ⋅         ⋅          ⋅
         ⋅          ⋅         ⋅          ⋅         …   ⋅         ⋅          ⋅
         ⋅          ⋅         ⋅          ⋅              ⋅         ⋅          ⋅
         ⋅          ⋅         ⋅          ⋅              ⋅         ⋅          ⋅
         ⋅          ⋅         ⋅          ⋅              ⋅         ⋅          ⋅
         ⋅          ⋅         ⋅          ⋅              ⋅         ⋅          ⋅
         ⋅          ⋅         ⋅          ⋅         …   ⋅         ⋅          ⋅
         ⋅          ⋅         ⋅          ⋅              ⋅         ⋅          ⋅
         ⋅          ⋅         ⋅          ⋅              ⋅         ⋅          ⋅
        ⋮                                          ⋱
         ⋅          ⋅         ⋅          ⋅              ⋅         ⋅          ⋅
         ⋅          ⋅         ⋅          ⋅              ⋅         ⋅          ⋅
         ⋅          ⋅         ⋅          ⋅         …   ⋅         ⋅          ⋅
         ⋅          ⋅         ⋅          ⋅              ⋅         ⋅          ⋅
         ⋅          ⋅         ⋅          ⋅              ⋅         ⋅          ⋅
         ⋅          ⋅         ⋅          ⋅              ⋅         ⋅          ⋅
         ⋅          ⋅         ⋅          ⋅              ⋅         ⋅          ⋅
         ⋅          ⋅         ⋅          ⋅         …   ⋅         ⋅          ⋅
         ⋅          ⋅         ⋅          ⋅            0.634171    ⋅          ⋅
         ⋅          ⋅         ⋅          ⋅            0.574875   1.15365     ⋅
         ⋅          ⋅         ⋅          ⋅            1.15365    0.994166  -0.232388
         ⋅          ⋅         ⋅          ⋅              ⋅       -0.232388   1.59481

[130]: @btime eigmax(A)

         851.990 ms (44 allocations: 183.11 MiB)

[130]: 6.4600741644987085
```

Рис. 4.7: Пятый раздел примеров (3).

Потом я выполнила примеры по общей линейной алгебре (рис. 4.8):

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

[132]: 3×3 Matrix{Rational{BigInt}}:
        7//10  7//10  2//5
        2//5   7//10  2//5
        3//5   3//5   9//10

[134]: x = fill(1, 3)

[134]: 3-element Vector{Int64}:
        1
        1
        1

[136]: b = Arational*x

[136]: 3-element Vector{Rational{BigInt}}:
         9//5
         3//2
        21//10

[138]: Arational\b

[138]: 3-element Vector{Rational{BigInt}}:
        1
        1
        1

[140]: lu(Arational)

[140]: LU{Rational{BigInt}, Matrix{Rational{BigInt}}, Vector{Int64}}
       L factor:
       3×3 Matrix{Rational{BigInt}}:
         1    0  0
        4//7  1  0
        6//7  0  1
       U factor:
       3×3 Matrix{Rational{BigInt}}:
        7//10  7//10   2//5
         0     3//10   6//35
         0      0     39//70
```

Рис. 4.8: Последний раздел примеров.

4.2 Задания для самостоятельной работы

Далее я приступила к выполнению заданий для самостоятельной работы.

4.2.1 Произведение векторов

Выполнила следующие задания (рис. 4.9):

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.

Произведение векторов

1. Задайте вектор v. Умножьте вектор v скалярно сам на себя и сохраните результат в dot_v.

[160]:
```
v = [1, 2, 3, 2, 1]
dot_v = dot(v, v)
```

[160]:
```
19
```

2. Умножьте v матрично на себя (внешнее произведение), присвоив результат переменной outer_v.

[162]:
```
outer_v = v * v'
```

[162]:
```
5×5 Matrix{Int64}:
 1  2  3  2  1
 2  4  6  4  2
 3  6  9  6  3
 2  4  6  4  2
 1  2  3  2  1
```

Рис. 4.9: Раздел 1, задания 1 и 2.

4.2.2 Системы линейных уравнений

Выполнила следующие задания:

1. Решить СЛАУ с двумя неизвестными (рис. 4.10-4.11):

1. Решить СЛАУ с двумя неизвестными.

[164]:
```
A = [1 1; 1 -1]
b = [2, 3]
x = A \ b
```

[164]:
```
2-element Vector{Float64}:
  2.5
 -0.5
```

[174]:
```
A = [1 1; 2 2]
b = [2, 4]
try
    x = A \ b
catch
    print("No solution")
end
```

```
No solution
```

[176]:
```
A = [1 1; 2 2]
b = [2, 5]
try
    x = A \ b
catch
    print("No solution")
end
```

```
No solution
```

Рис. 4.10: Раздел 2, задание 1 (1).

[178]:
```
A = [1 1; 2 2; 3 3]
b = [1, 2, 3]
try
    x = A \ b
catch
    print("No solution")
end
```

[178]:
```
2-element Vector{Float64}:
 0.4999999999999999
 0.5
```

[180]:
```
A = [1 1; 2 1; 1 -1]
b = [2, 1, 3]
try
    x = A \ b
catch
    print("No solution")
end
```

[180]:
```
2-element Vector{Float64}:
  1.5000000000000004
 -0.9999999999999997
```

[182]:
```
A = [1 1; 2 1; 3 2]
b = [2, 1, 3]
try
    x = A \ b
catch
    print("No solution")
end
```

[182]:
```
2-element Vector{Float64}:
 -0.9999999999999989
  2.9999999999999982
```

Рис. 4.11: Раздел 2, задание 1 (2).

2. Решить СЛАУ с двумя неизвестными (рис. 4.12):

2. Решить СЛАУ с тремя неизвестными.

[184]:
```
A = [1 1 1; 1 -1 -2]
b = [2, 3]
try
    x = A \ b
catch
    print("No solution")
end
```

[184]:
```
3-element Vector{Float64}:
  2.2142857142857144
  0.35714285714285704
 -0.5714285714285712
```

[192]:
```
A = [1 1 1; 2 2 -3; 3 1 1]
b = [2, 4, 1]
try
    x = A \ b
catch
    print("No solution")
end
```

[192]:
```
3-element Vector{Float64}:
 -0.5
  2.5
  0.0
```

[210]:
```
A = [1 1 1; 1 1 2; 2 2 3]
b = [1, 0, 1]
pinv(A) * b
#rank([A b]), rank(A)
```

[210]:
```
3-element Vector{Float64}:
  0.9999999999999994
  1.0000000000000016
 -1.0000000000000007
```

[212]:
```
A = [1 1 1; 1 1 2; 2 2 3]
b = [1, 0, 0]
#pinv(A) * b
rank([A b]), rank(A)
```

[212]:
```
(3, 2)
```

Рис. 4.12: Раздел 2, задание 2.

4.2.3 Операции с матрицами

Выполнила следующие задания:

1. Приведите приведённые ниже матрицы к диагональному виду (рис. 4.13).

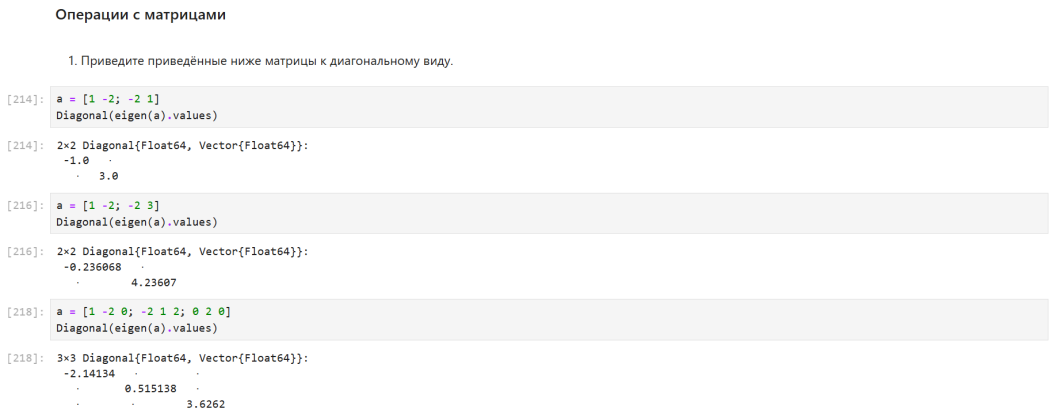

Рис. 4.13: Раздел 3, задание 1.

2. Вычислите (рис. 4.14):

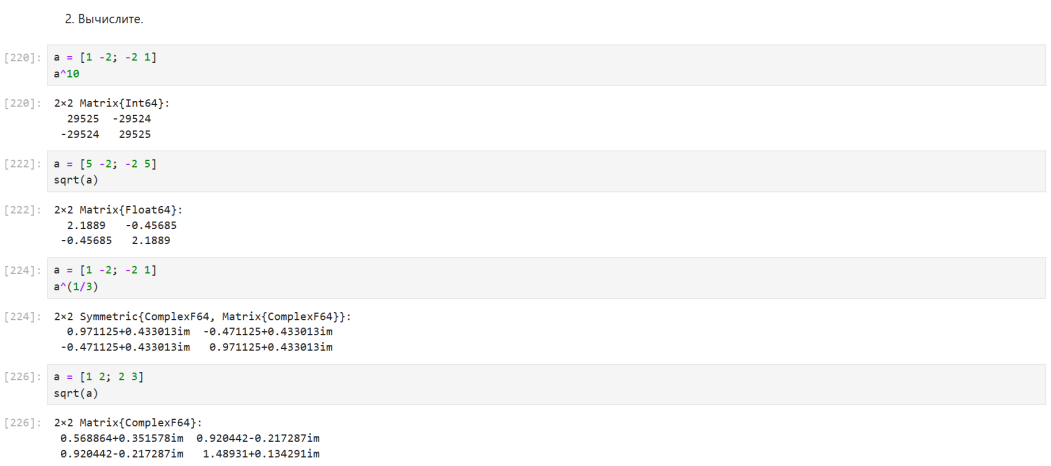

Рис. 4.14: Раздел 3, задание 2.

3. Найдите собственные значения матрицы A. Создайте диагональную матрицу из собственных значений матрицы A. Создайте нижнедиагональную матрицу из матрицы A. Оцените эффективность выполняемых операций (рис. 4.15).

3. Найдите собственные значения матрицы A. Создайте диагональную матрицу из собственных значений матрицы A. Создайте нижнедиагональную матрицу из матрицы A. Оцените эффективность выполняемых операций.

```
[228]: 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];

[232]: @time self = eigen(A)

  0.000106 seconds (19 allocations: 3.047 KiB)

[232]: 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

[236]: @time d = Diagonal(self.values)

  0.000025 seconds (1 allocation: 16 bytes)

[236]: 5×5 Diagonal{Float64, Vector{Float64}}:
 -128.493      ⋅         ⋅         ⋅         ⋅
     ⋅      -55.8878     ⋅         ⋅         ⋅
     ⋅         ⋅      42.7522      ⋅         ⋅
     ⋅         ⋅         ⋅      87.1611      ⋅
     ⋅         ⋅         ⋅         ⋅      542.468

[238]: @time ld = LowerTriangular(A)

  0.014404 seconds (81 allocations: 3.828 KiB, 99.58% compilation time)

[238]: 5×5 LowerTriangular{Int64, Matrix{Int64}}:
 140    ⋅    ⋅    ⋅    ⋅
  97  106    ⋅    ⋅    ⋅
  74   89  152    ⋅    ⋅
 168  131  144   54    ⋅
 131   36   71  142   36
```

Рис. 4.15: Раздел 3, задание 3.

4.2.4 Линейные модели экономики

1. Матрица A называется продуктивной, если решение x системы при любой неотрицательной правой части y имеет только неотрицательные элементы x_i. Используя это определение, проверьте, являются ли матрицы продуктивными (рис. 4.16):

Линейные модели экономики

1. Матрица A называется продуктивной, если решение x системы при любой неотрицательной правой части y имеет только неотрицательные элементы x_i. Используя это определение, проверьте, являются ли матрицы продуктивными.

```
[242]: a = [1 2; 3 4]
       all(>=(0), inv(I - a)) ? "Продуктивная" : "Непродуктивная"

[242]: "Непродуктивная"

[244]: a = 0.5 * [1 2; 3 4]
       all(>=(0), inv(I - a)) ? "Продуктивная" : "Непродуктивная"

[244]: "Непродуктивная"

[246]: a = 0.1 * [1 2; 3 4]
       all(>=(0), inv(I - a)) ? "Продуктивная" : "Непродуктивная"

[246]: "Продуктивная"
```

Рис. 4.16: Раздел 4, задание 1.

2. Критерий продуктивности: матрица A является продуктивной тогда и только тогда, когда все элементы матрица $(E - A)^{-1}$ являются неотрицательными числами. Используя этот критерий, проверьте, являются ли матрицы продуктивными (рис. 4.17):

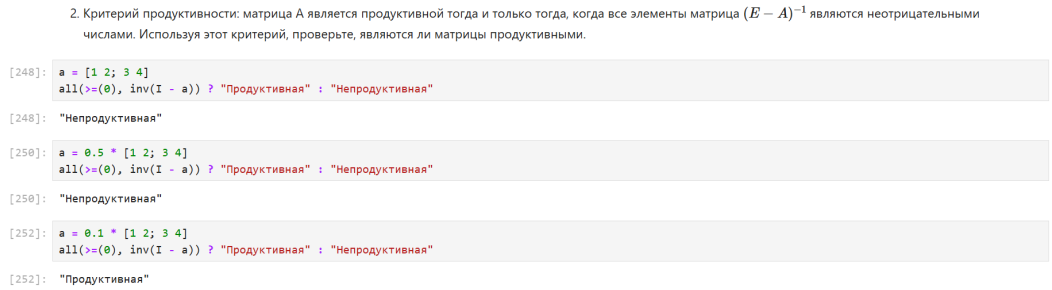

Рис. 4.17: Раздел 4, задание 2.

3. Спектральный критерий продуктивности: матрица A является продуктивной тогда и только тогда, когда все её собственные значения по модулю меньше 1. Используя этот критерий, проверьте, являются ли матрицы продуктивными (рис. 4.18):

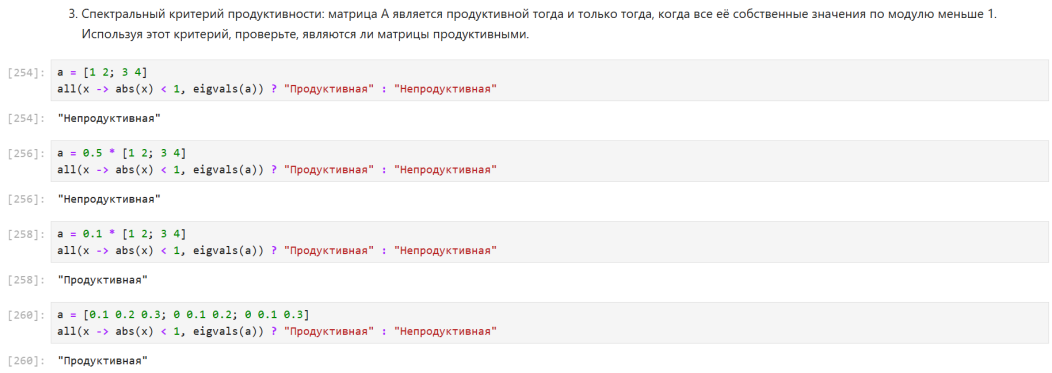

Рис. 4.18: Раздел 4, задание 3.

5 Выводы

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