Лабораторная работа 4

Линейная алгебра

Ланцова Яна Игоревна

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

1. Используя JupyterLab, повторим примеры.
2. Выполним задания для самостоятельной работы.

3 Выполнение лабораторной работы

Выполним примеры из раздела про поэлементные операции над многомерными массивами (рис. 3.1).

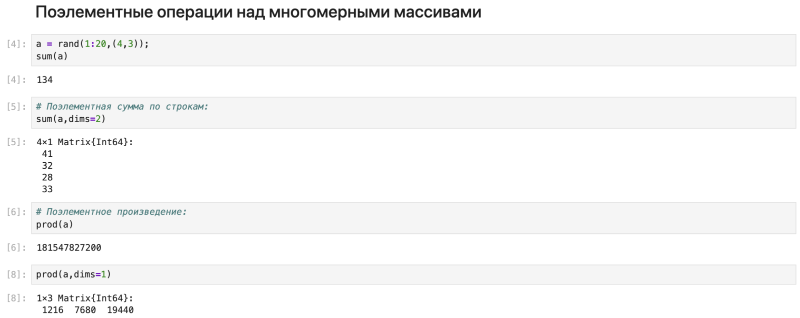


Рис. 3.1: Поэлементные операции над многомерными массивами

Для работы со средними значениями можно воспользоваться возможностями пакета Statistics (рис. 3.2).

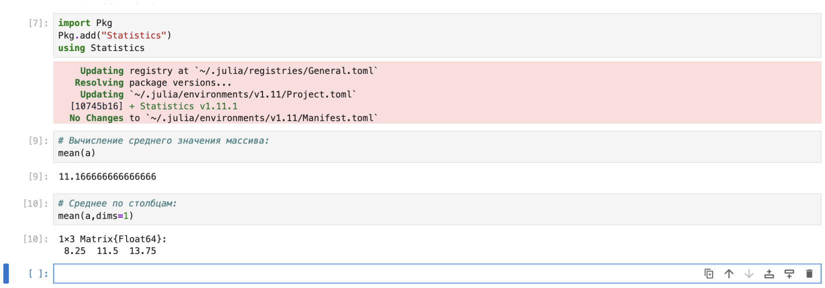


Рис. 3.2: Поэлементные операции над многомерными массивами

Выполним примеры из раздела про транспонирование,след,ранг,определительи инверсия матрицы (рис. 3.3).

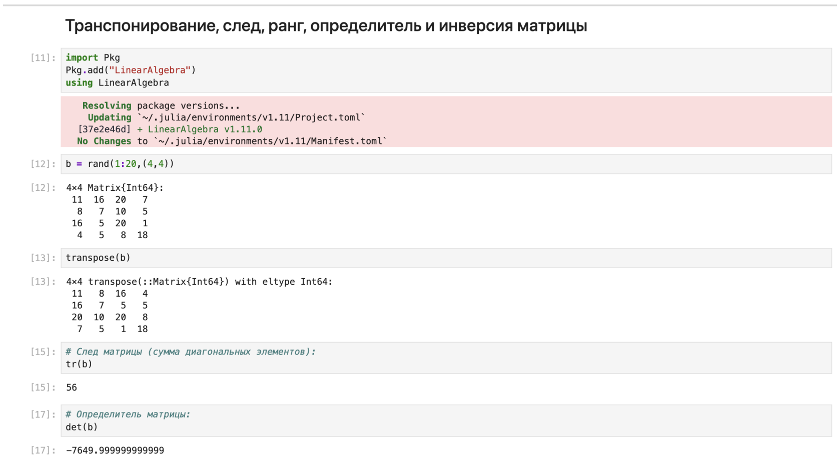


Рис. 3.3: Транспонирование, след, ранг, определительи инверсия матрицы

Выполним примеры из раздела про вычисление нормы векторов и матриц, повороты, вращения (рис. 3.4).

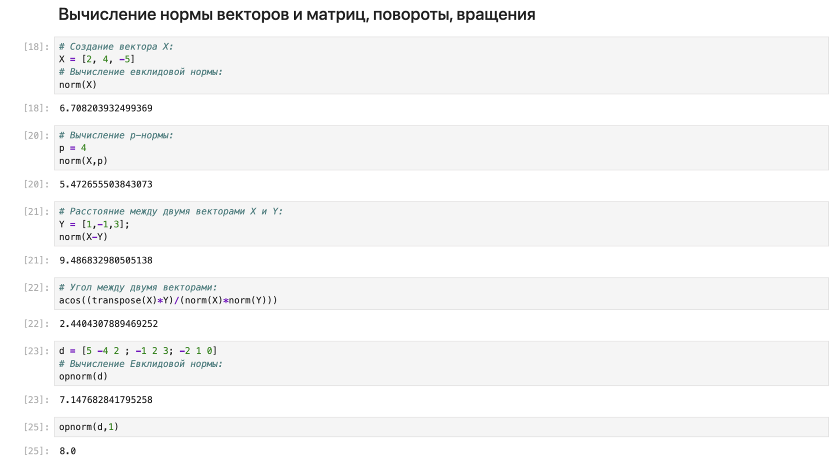


Рис. 3.4: Вычисление нормы векторов и матриц

Выполним примеры из раздела про матричное умножение, единичная матрица, скалярное произведение и примеры из раздела про факторизацию и специальные матричные структуры (рис. 3.5).

Матричное умножение, единичная матрица, скалярное произведение

```
[26]: A = rand(1:10,(2,3))
      B = rand(1:10,(3,4))
      # Произведение матриц A и B:
      A*B

[26]: 2×4 Matrix{Int64}:
       38  27  39   63
       64  47  69  101

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

[27]: -17
```

Факторизация. Специальные матричные структуры

```
[28]: A = rand(3, 3)
      x = fill(1.0, 3)
      b = A*x
      # Решение исходного уравнения получаем с помощью функции \
      # (убеждаемся, что x - единичный вектор):
      A\b

[28]: 3-element Vector{Float64}:
       0.9999999999999999
       1.0
       1.0000000000000002

[29]: # LU-факторизация:
      Alu = lu(A)

[29]: LU{Float64, Matrix{Float64}, Vector{Int64}}
      L factor:
```

Рис. 3.5: Факторизация. Специальные матричные структуры

Для оценки эффективности выполнения операций над матрицами большой размерности и специальной структуры воспользуемся пакетом BenchmarkTools (рис. 3.6).

```
[34]: Asym = A + A'
      Asym_noisy = copy(Asym)
      Asym_noisy[1,2] += 5eps();

[35]: import Pkg
      Pkg.add("BenchmarkTools")
      using BenchmarkTools
      # Оценка эффективности выполнения операции по нахождению
      # собственных значений симметризованной матрицы:
      @btime eigvals(Asym);
      # Оценка эффективности выполнения операции по нахождению
      # собственных значений зашумлённой матрицы:
      @btime eigvals(Asym_noisy);

         Resolving package versions...
        No Changes to `~/.julia/environments/v1.11/Project.toml`
        No Changes to `~/.julia/environments/v1.11/Manifest.toml`
        725.063 ns (15 allocations: 1.66 KiB)
        1.042 μs (17 allocations: 1.48 KiB)

[36]: # Трёхдиагональная матрица 1000000 x 1000000:
      n = 1000000;
      A = SymTridiagonal(randn(n), randn(n-1))
      # Оценка эффективности выполнения операции по нахождению собственных значений:
      @btime eigmax(A)

        370.908 ms (44 allocations: 183.24 MiB)

[36]: 6.229002503394929
```

Рис. 3.6: Факторизация. Специальные матричные структуры

Выполним примеры из раздела про общую линейную алгебру (рис. 3.7).

Общая линейная алгебра

```
[37]: # Матрица с рациональными элементами:
Arational = Matrix{Rational{BigInt}}(rand(1:10, 3, 3))/10
# Единичный вектор:
x = fill(1, 3)

[37]: 3-element Vector{Int64}:
 1
 1
 1

[38]: # Задаём вектор b:
b = Arational*x

[38]: 3-element Vector{Rational{BigInt}}:
  8//5
 19//10
  6//5

[39]: # Решение исходного уравнения получаем с помощью функции (убеждаемся, что x – единичный вектор):
Arational\b
# LU-разложение:
lu(Arational)

[39]: LU{Rational{BigInt}, Matrix{Rational{BigInt}}, Vector{Int64}}
L factor:
3×3 Matrix{Rational{BigInt}}:
  1      0    0
 3//5    1    0
 3//5  -3//7  1
U factor:
3×3 Matrix{Rational{BigInt}}:
 1   4//5    1//10
 0  -7//25  17//50
 0    0     17//35
```

Рис. 3.7: Общая линейная алгебра

Теперь перейдем к выполнению заданий для самостоятельной работы.

3.1 Задание 1

Зададим вектор v. Умножим вектор v скалярно сам на себя и сохраним результат в dot_v. Также умножим v матрично на себя (внешнее произведение), присвоив результат переменной outer_v (рис. 3.8).

Задания для самостоятельного выполнения

Задание 1. Произведение векторов

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

```
[41]: v = [6, 2, 7, 4, 5]

[41]: 5-element Vector{Int64}:
 6
 2
 7
 4
 5

[42]: dot_v = dot(v, v)

[42]: 130
```

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

```
[44]: outer_v = v * transpose(v)

[44]: 5×5 Matrix{Int64}:
 36  12  42  24  30
 12   4  14   8  10
 42  14  49  28  35
 24   8  28  16  20
 30  10  35  20  25
```

Рис. 3.8: Произведение векторов

3.2 Задание 2

- Решим СЛАУ с двумя неизвестными (рис. 3.9 - 3.10).

Задание 2. Системы линейных уравнений

Решить СЛАУ с двумя неизвестными.

```
[45]: A_a = [1 1; 1 -1]
      b_a = [2, 3]
      x = A_a \ b_a

[45]: 2-element Vector{Float64}:
        2.5
       -0.5

[47]: A_b = [1 1; 2 2]
      b_b = [2, 4]
      if (det(A_b)!=0)
          print(A_b \ b_b)
      else
          print("No solution")
      end

      No solution

[51]: A_c = [1 1; 2 2]
      b_c = [2, 5]
      if (det(A_c)!=0)
          print(A_c \ b_c)
      else
          print("No solution")
      end

      No solution
```

Рис. 3.9: Системы линейных уравнений

```
[49]: A_d = [1 1; 2 2; 3 3]
      b_d = [1, 2, 3]
      print(A_d \ b_d)

      [0.5, 0.4999999999999997]

[50]: A_e = [1 1; 2 1; 1 -1]
      b_e = [2, 1, 3]
      print(A_e \ b_e)

      [1.5000000000000004, -0.9999999999999996]

[55]: A_f = [1 1; 2 1; 3 2]
      b_f = [2, 1, 3]
      print(A_f \ b_f)

      [-1.0000000000000009, 3.0000000000000018]

[ ]:
```

Рис. 3.10: Системы линейных уравнений

- Решим СЛАУ с тремя неизвестными (рис. 3.11).

Решить СЛАУ с тремя неизвестными.

```
[56]: A_a = [1 1 1; 1 -1 -2]
      b_a = [2, 3]
      # Решение через псевдообратную матрицу
      x_a = pinv(A_a) * b_a

[56]: 3-element Vector{Float64}:
        2.214285714285714
        0.35714285714285676
       -0.5714285714285716

[57]: A_b = [1 1 1; 2 2 -3; 3 1 1]
      b_b = [2, 4, 1]
      x_b = pinv(A_b) * b_b

[57]: 3-element Vector{Float64}:
       -0.4999999999999999
        2.5
        3.885780586188048e-16

[58]: A_c = [1 1 1; 1 1 2; 2 2 3]
      b_c = [1, 0, 1]
      x_c = pinv(A_c) * b_c

[58]: 3-element Vector{Float64}:
        0.9999999999999991
        1.0000000000000018
       -1.0000000000000002

[59]: A_d = [1 1 1; 1 1 2; 2 2 3]
      b_d = [1, 0, 0]
      x_d = pinv(A_d) * b_d

[59]: 3-element Vector{Float64}:
        0.8333333333333323
        0.8333333333333346
       -0.9999999999999999
```

Рис. 3.11: Системы линейных уравнений

3.3 Задание 3

Приведем матрицы к диагональному виду (рис. 3.12):

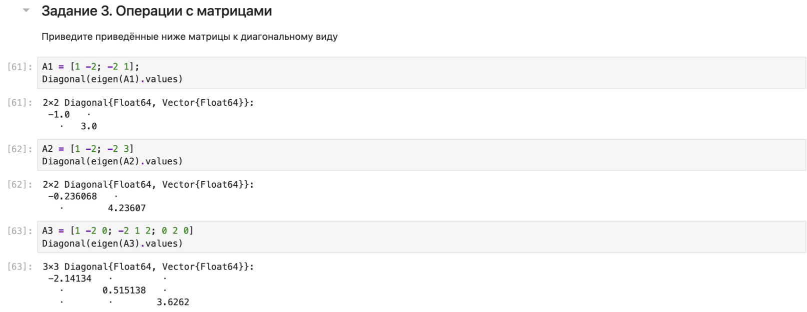


Рис. 3.12: Операции с матрицами

Вычислим следующие операции с матрицами (рис. 3.13):

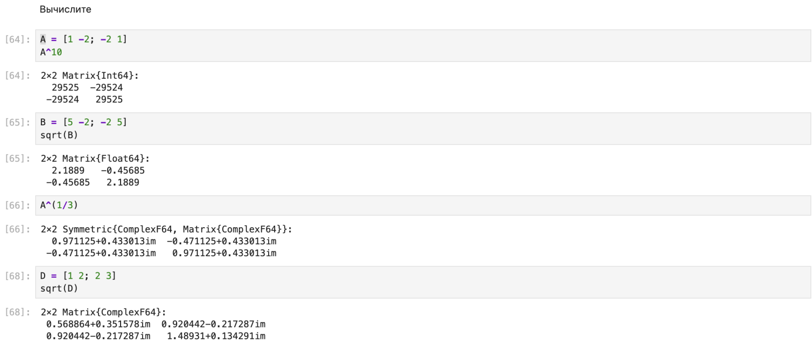
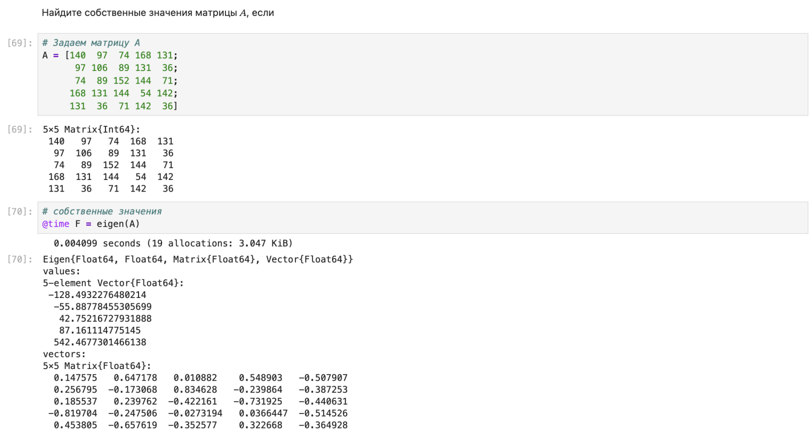


Рис. 3.13: Операции с матрицами

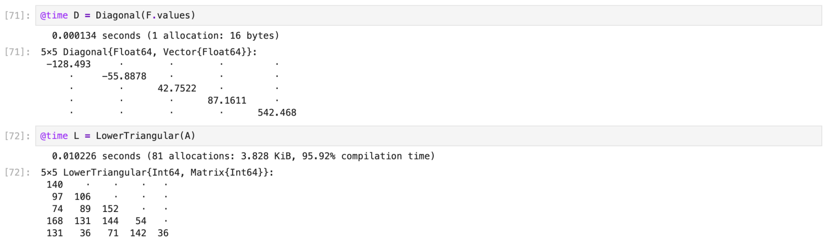
Найдем собственные значения матрицы А. Создадим диагональную матрицу из собственных значений матрицы А. Создадим нижнедиагональную матрицу из матрицы А. Оценим эффективность выполняемых операций (рис. 3.14 - 3.15):



Найдите собственные значения матрицы A, если

```
[69]: # Задаем матрицу A
      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]

[69]: 5×5 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

[70]: # собственные значения
      @time F = eigen(A)

        0.004099 seconds (19 allocations: 3.047 KiB)

[70]: Eigen{Float64, Float64, Matrix{Float64}, Vector{Float64}}
      values:
      5-element Vector{Float64}:
       -128.4932276480214
        -55.88778455305699
         42.75216727931888
         87.161114775145
        542.4677301466138
      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
```

Рис. 3.14: Операции с матрицами



```
[71]: @time D = Diagonal(F.values)

        0.000134 seconds (1 allocation: 16 bytes)

[71]: 5×5 Diagonal{Float64, Vector{Float64}}:
       -128.493     ⋅         ⋅         ⋅         ⋅
           ⋅     -55.8878     ⋅         ⋅         ⋅
           ⋅         ⋅      42.7522     ⋅         ⋅
           ⋅         ⋅         ⋅      87.1611     ⋅
           ⋅         ⋅         ⋅         ⋅      542.468

[72]: @time L = LowerTriangular(A)

        0.010226 seconds (81 allocations: 3.828 KiB, 95.92% compilation time)

[72]: 5×5 LowerTriangular{Int64, Matrix{Int64}}:
       140    ⋅    ⋅    ⋅   ⋅
        97  106    ⋅    ⋅   ⋅
        74   89  152    ⋅   ⋅
       168  131  144   54   ⋅
       131   36   71  142  36
```

Рис. 3.15: Оценка эффективности

3.4 Задание 4

Линейная модель может быть записана как СЛАУ

$$x - Ax = y$$

где элементы матрицы A и столбца y – неотрицательные числа. По своему смыслу в экономике элементы матрицы A и столбцов x, y не могут быть отрицательными числами. Матрица A называется продуктивной,если решение x системы при любой неотрицательной правой части y имеет только неотрицательные элементы x_i.Используя это определение, проверим, являются ли матрицы родуктивными (рис. 3.16).

Задание 4. Линейные модели экономики

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

```
[75]: A1 = [1 2; 3 4]
      inv_IA1 = inv(I - A1)

[75]: 2×2 Matrix{Float64}:
        0.5  -0.333333
       -0.5   0.0

[76]: all(>=(0), inv_IA1) ? "Да" : "НЕТ"

[76]: "НЕТ"

[77]: A2 = 0.5 * [1 2; 3 4]
      inv_IA2 = inv(I - A2)

[77]: 2×2 Matrix{Float64}:
        0.5   -0.5
       -0.75  -0.25

[78]: all(>=(0), inv_IA2) ? "Да" : "НЕТ"

[78]: "НЕТ"

[79]: A3 = 0.1 * [1 2; 3 4]
      inv_IA3 = inv(I - A3)

[79]: 2×2 Matrix{Float64}:
       1.25   0.416667
       0.625  1.875

[80]: all(>=(0), inv_IA3) ? "Да" : "НЕТ"

[80]: "Да"
```

Рис. 3.16: Линейные модели экономики

Критерий продуктивности: матрица А является продуктивной тогда и только тогда, когда все элементы матрицы

$$(E - A)^-1$$

являются неотрицательными числами. Используя этот критерий, проверим, являются ли матрицы продуктивными (рис. 3.17).

Проверьте, являются ли матрицы продуктивными.

```
[84]: inv_EA1 = inv(I - A1)

[84]: 2×2 Matrix{Float64}:
       -0.0  -0.333333
       -0.5   0.0

[83]: all(>=(0), inv_EA1) ? "Да" : "НЕТ"

[83]: "НЕТ"

[85]: inv_EA2 = inv(I - A2)

[85]: 2×2 Matrix{Float64}:
        0.5   -0.5
       -0.75  -0.25

[86]: all(>=(0), inv_EA2) ? "Да" : "НЕТ"

[86]: "НЕТ"

[87]: inv_EA3 = inv(I - A3)

[87]: 2×2 Matrix{Float64}:
       1.25   0.416667
       0.625  1.875

[88]: all(>=(0), inv_EA3) ? "Да" : "НЕТ"

[88]: "Да"
```

Рис. 3.17: Линейные модели экономики

Спектральный критерий продуктивности: матрица А является продуктивной тогда и только тогда, когда все её собственные значения по модулю меньше 1. Используя этот критерий, проверим, являются ли матрицы продуктивными (рис. 3.18).

Рис. 3.18: Линейные модели экономики

4 Выводы

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