# Лабораторная работа № 4

**Линейная алгебра**

Сунгурова Мариян

# Содержание

# Список иллюстраций

# 1 Цель работы

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

# 2 Задание

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

# 3 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].

# 4 Выполнение лабораторной работы

Выполним примеры из раздела про поэлементные операции над многомерными массивами (рис. 4.1-4.2).

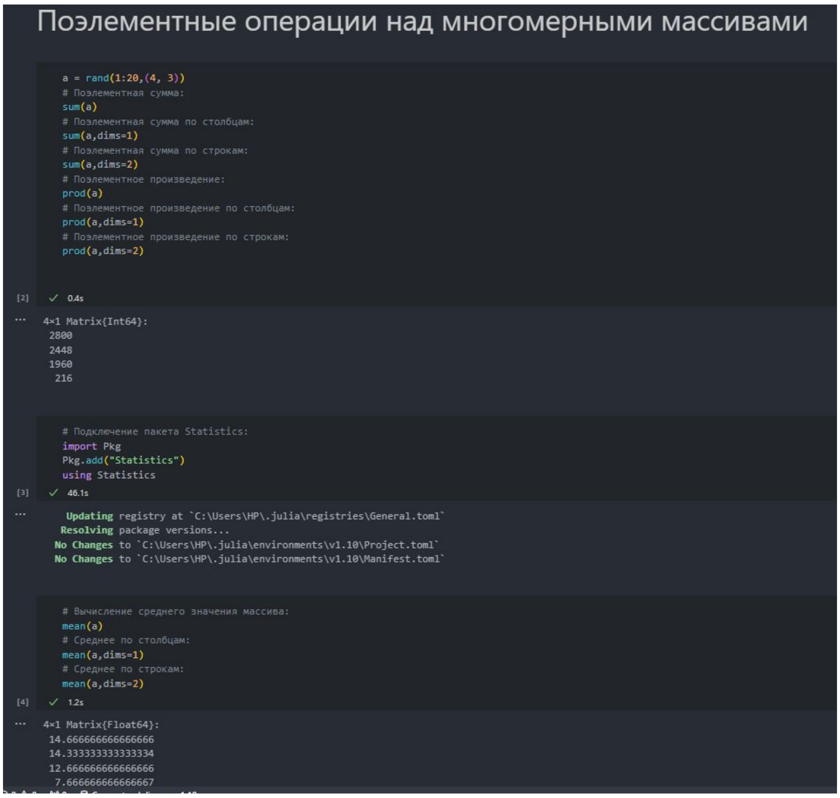



Рис. 4.1: Поэлементные операции над многомерными массивами

Выполним примеры из раздела про транспонирование,след,ранг,определитель и инверсия матрицы (рис. 4.3).

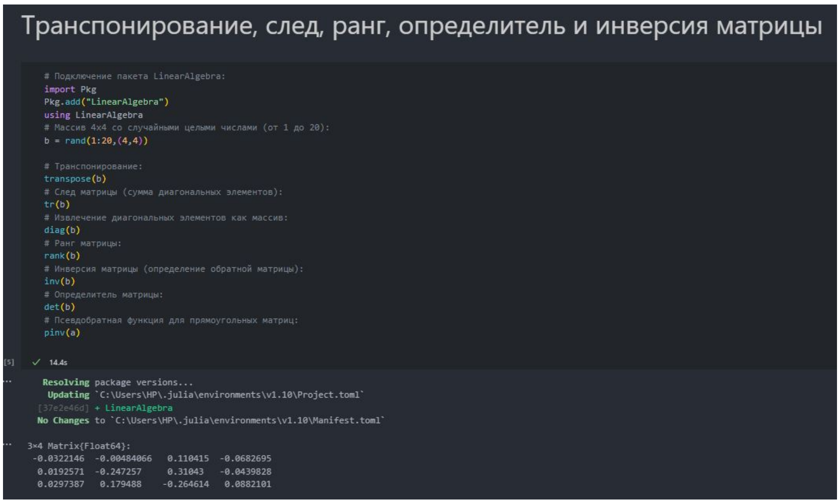



Рис. 4.2: Транспонирование,след,ранг,определительи инверсия матрицы

Выполним примеры из раздела про вычисление нормы векторов и матриц, повороты и вращения (рис. 4.3 - 4.4).

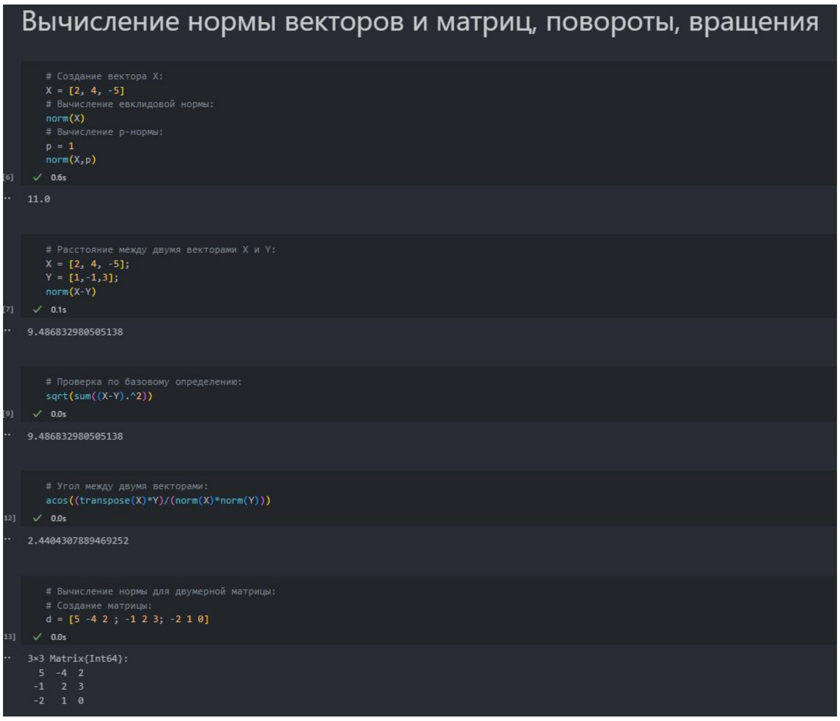



Рис. 4.3: Нормы векторов и матриц, повороты и вращения

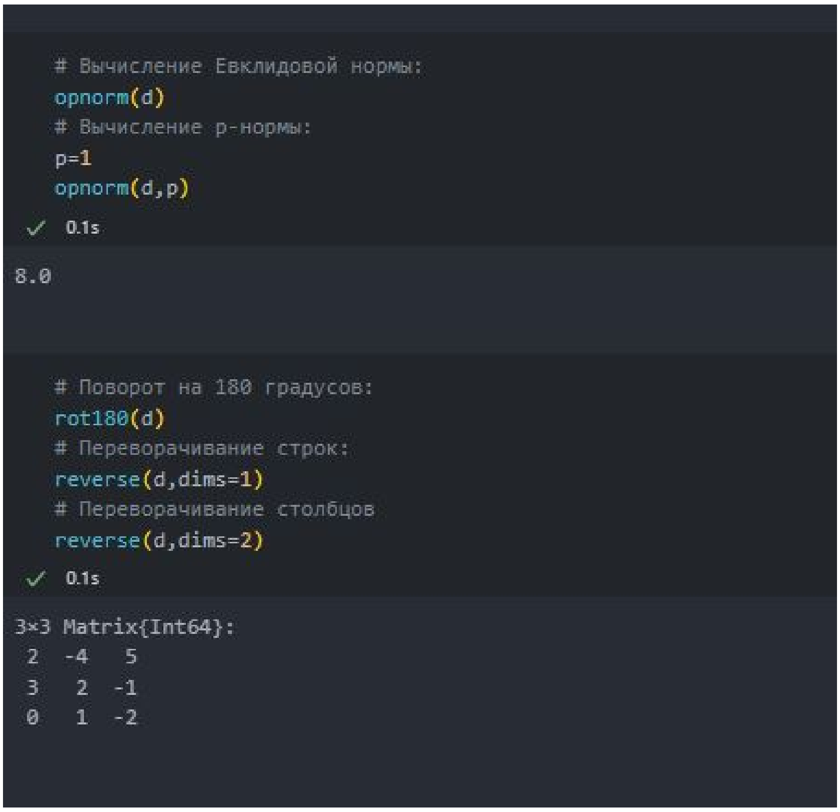



Рис. 4.4: Нормы векторов и матриц, повороты и вращения

Выполним примеры из раздела про матричное умножение,единичная матрица,скалярное произведение (рис. 4.5).

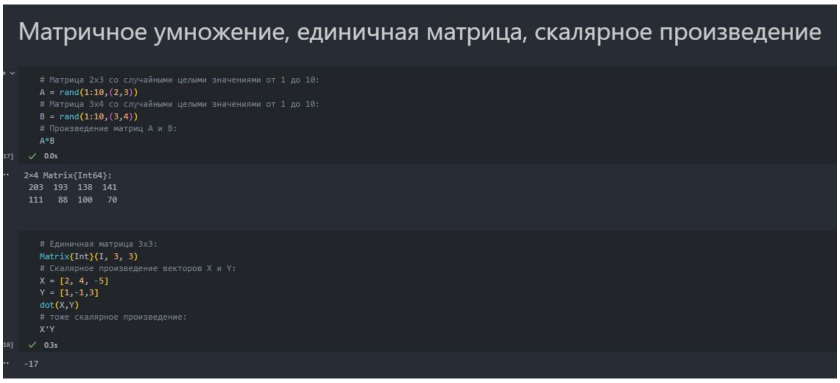
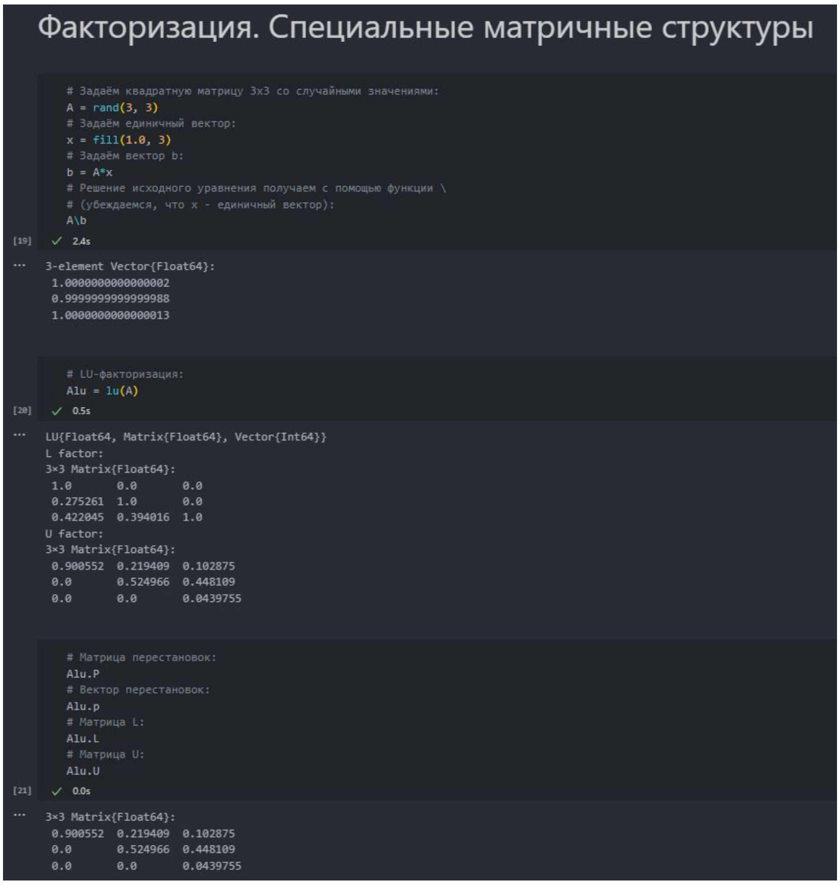



Рис. 4.5: Матричное умножение,единичная матрица,скалярное произведение

Выполним примеры из раздела про факторизацию и специальные матричные структуры (рис. 4.6-4.8).



Факторизация. Специальные матричные структуры

```
# Задаём квадратную матрицу 3x3 со случайными значениями:
A = rand(3, 3)
# Задаём единичный вектор:
x = fill(1.0, 3)
# Задаём вектор b:
b = A*x
# Решение исходного уравнения получаем с помощью функции \
# (убеждаемся, что x - единичный вектор):
A\b
```

[19] ✓ 2.4s

```
3-element Vector{Float64}:
 1.0000000000000002
 0.9999999999999988
 1.0000000000000013
```

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

[20] ✓ 0.5s

```
LU{Float64, Matrix{Float64}, Vector{Int64}}
L factor:
3×3 Matrix{Float64}:
 1.0       0.0       0.0
 0.275261  1.0       0.0
 0.422045  0.394016  1.0
U factor:
3×3 Matrix{Float64}:
 0.900552  0.219409  0.102875
 0.0       0.524966  0.448109
 0.0       0.0       0.0439755
```

```
# Матрица перестановок:
Alu.P
# Вектор перестановок:
Alu.p
# Матрица L:
Alu.L
# Матрица U:
Alu.U
```

[21] ✓ 0.0s

```
3×3 Matrix{Float64}:
 0.900552  0.219409  0.102875
 0.0       0.524966  0.448109
 0.0       0.0       0.0439755
```

Рис. 4.6: Факторизация.Специальные матричные структуры

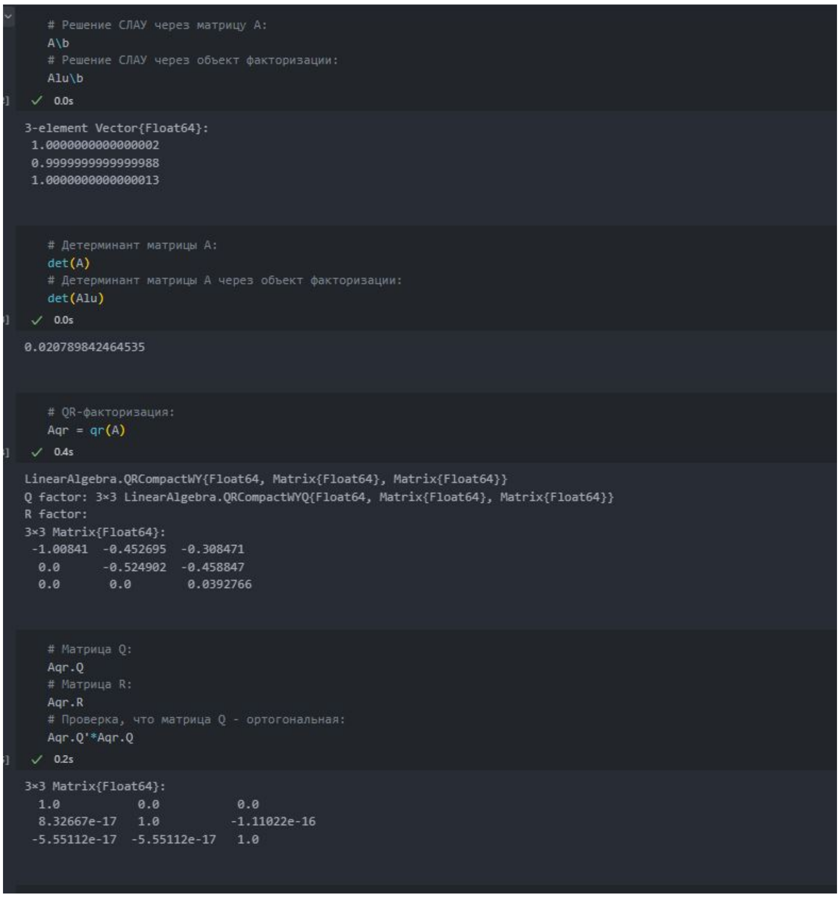



Рис. 4.7: Факторизация.Специальные матричные структуры

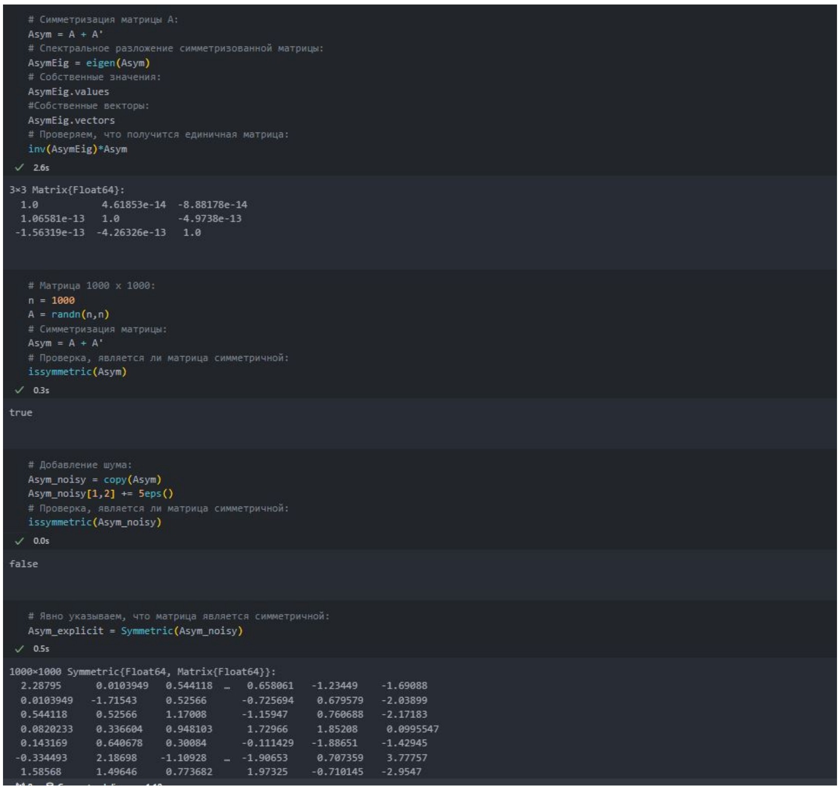



Рис. 4.8: Факторизация.Специальные матричные структуры

Далее для оценки эффективности выполнения операций над матрицами большой размерности и специальной структуры воспользуемся пакетом BenchmarkTools

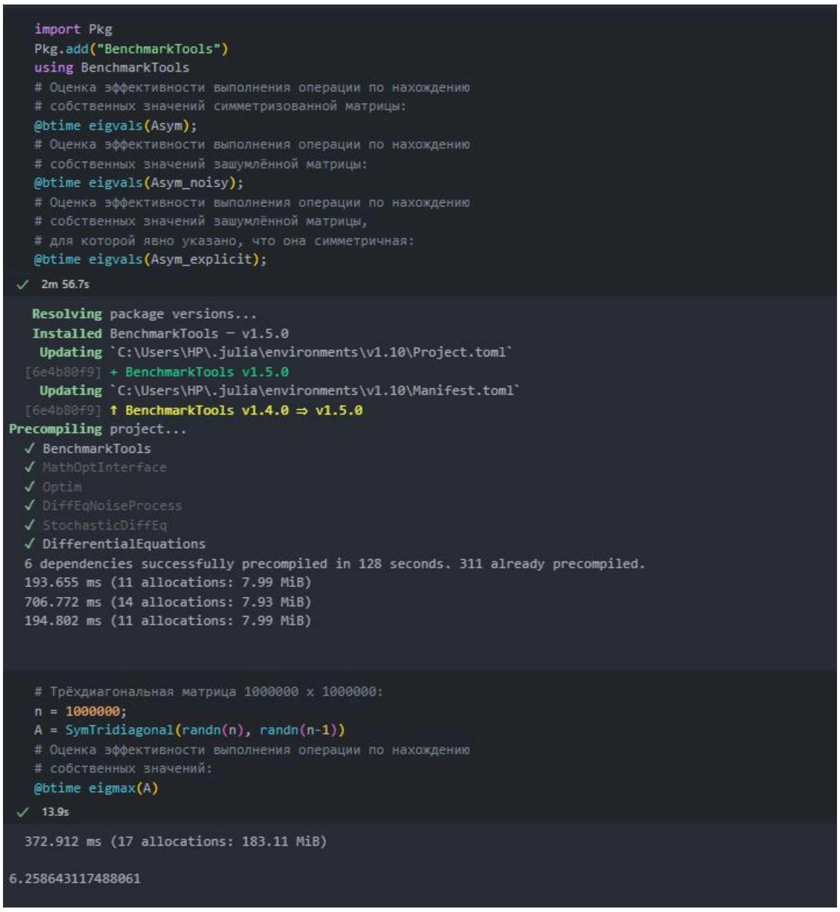



Рис. 4.9: Факторизация.Специальные матричные структуры

Выполним примеры из раздела про общую линейную алгебру (рис. 4.10).

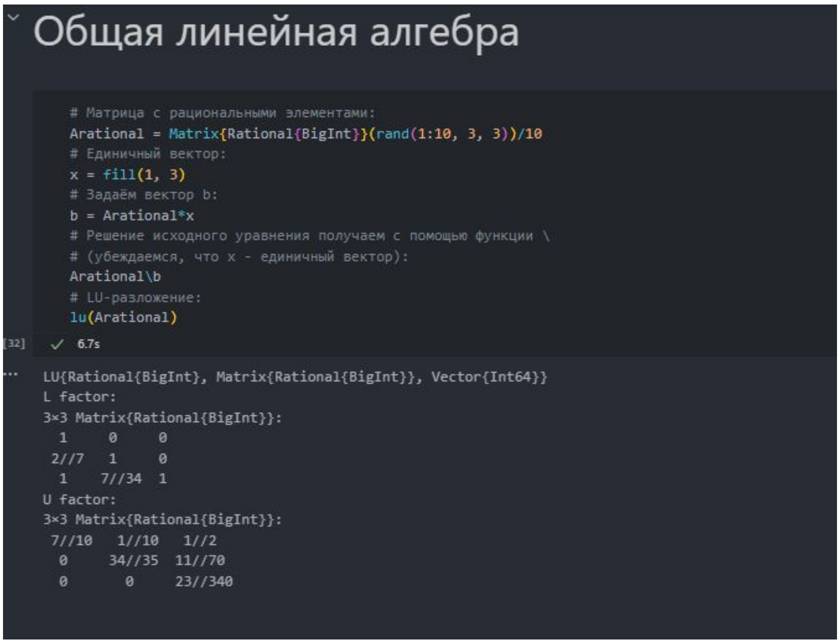



Рис. 4.10: Общаялинейная алгебра

## 4.1 Заданиядля самостоятельного выполнения

Зададим вектор v. Умножим вектор v скалярно сам на себя и сохраним результат bdot_v Затем умножим v матрично на себя(внешнее произведение), присвоив результат переменной outer_v .(рис. 4.11).

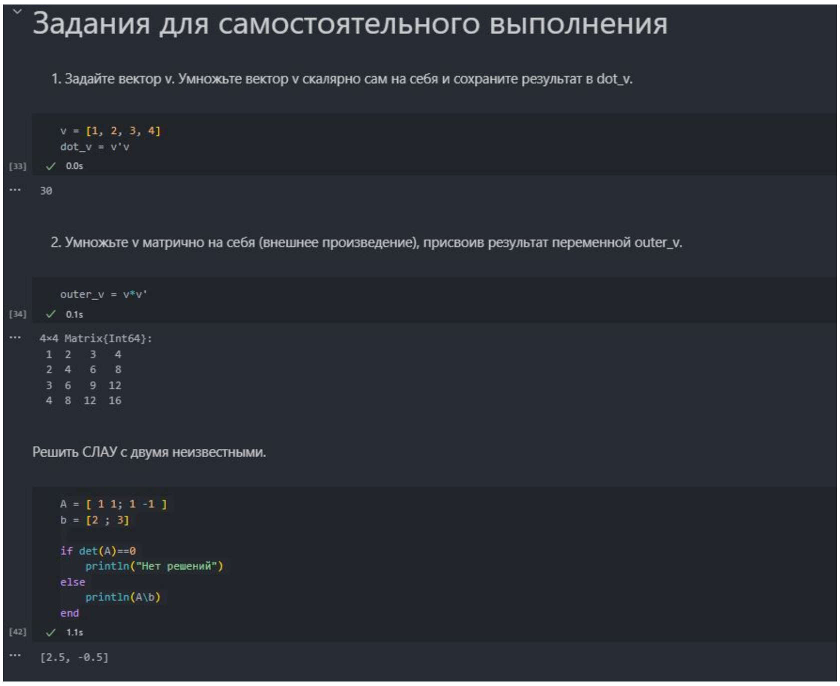



Рис. 4.11: Произведение векторов

Решим СЛАУ с двумя неизвестными (рис. 4.12-4.13).

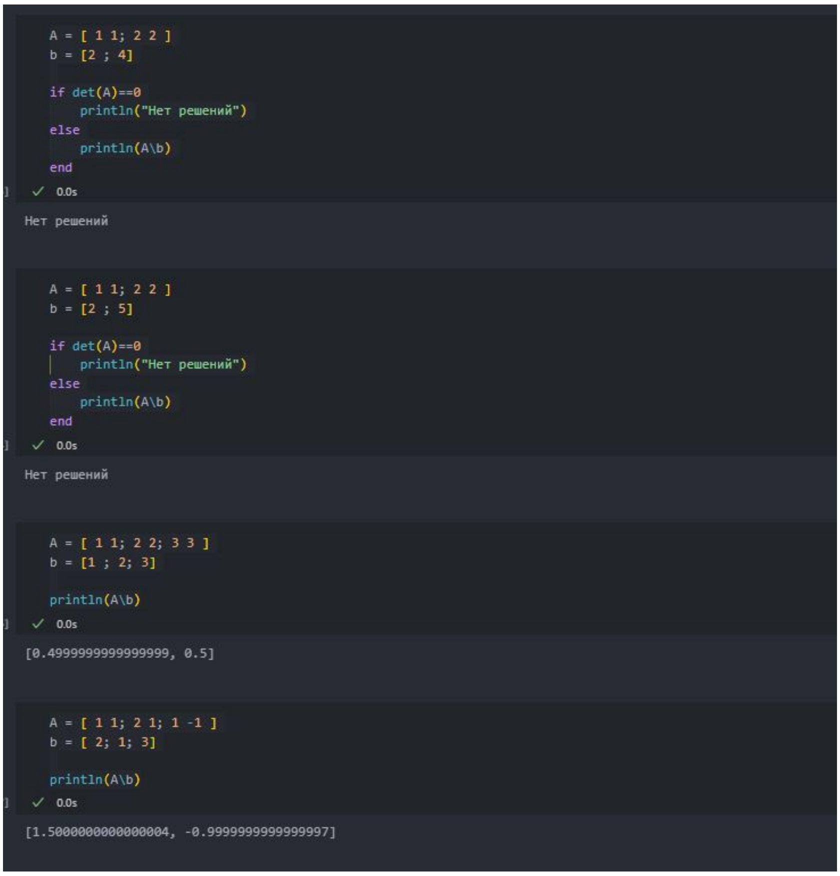



Рис. 4.12: Системы линейных уравнений

```
A = [ 1 1; 2 1; 3 2 ]
b = [ 2; 1; 3]

println(A\b)
```
✓ 0.0s

```
[-0.9999999999999989, 2.9999999999999982]
```

Решить СЛАУ с тремя неизвестными.

```
A = [ 1 1 1; 1 -1 -2 ]
b = [ 2; 3 ]

println(A\b)
```
✓ 0.0s

```
[2.2142857142857144, 0.35714285714285704, -0.5714285714285712]
```

```
A = [ 1 1 1; 2 2 -3; 3 1 1 ]
b = [ 2; 4; 1]

println(A\b)
```
✓ 0.0s

```
[-0.5, 2.5, 0.0]
```

```
A = [ 1 1 1; 1 1 2; 2 2 3 ]
b = [ 1; 0; 1]
if det(A)==0
    println("Нет решений")
else
    println(A\b)
end
```
✓ 0.0s

```
Нет решений
```

```
A = [ 1 1 1; 1 1 2; 2 2 3 ]
b = [ 1; 0; 0]
if det(A)==0
    println("Нет решений")
else
    println(A\b)
end
```
✓ 0.0s

Рис. 4.13: Систем линейных уравнений

Решим СЛАУ с тремя неизвестными (рис. 4.14).

```julia
A = [ 1 1; 2 1; 3 2 ]
b = [ 2; 1; 3]

println(A\b)
```

0.0s

```
[-0.9999999999999989, 2.9999999999999982]
```

Решить СЛАУ с тремя неизвестными.

```julia
A = [ 1 1 1; 1 -1 -2 ]
b = [ 2; 3 ]

println(A\b)
```

0.0s

```
[2.2142857142857144, 0.35714285714285704, -0.5714285714285712]
```

```julia
A = [ 1 1 1; 2 2 -3; 3 1 1 ]
b = [ 2; 4; 1]

println(A\b)
```

0.0s

```
[-0.5, 2.5, 0.0]
```

```julia
A = [ 1 1 1; 1 1 2; 2 2 3 ]
b = [ 1; 0; 1]
if det(A)==0
    println("Нет решений")
else
    println(A\b)
end
```

0.0s

```
Нет решений
```

```julia
A = [ 1 1 1; 1 1 2; 2 2 3 ]
b = [ 1; 0; 0]
if det(A)==0
    println("Нет решений")
else
    println(A\b)
end
```

0.0s

Рис. 4.14: Систем линейных уравнений

Приведем матрицы к диагональному виду (рис. 4.15).

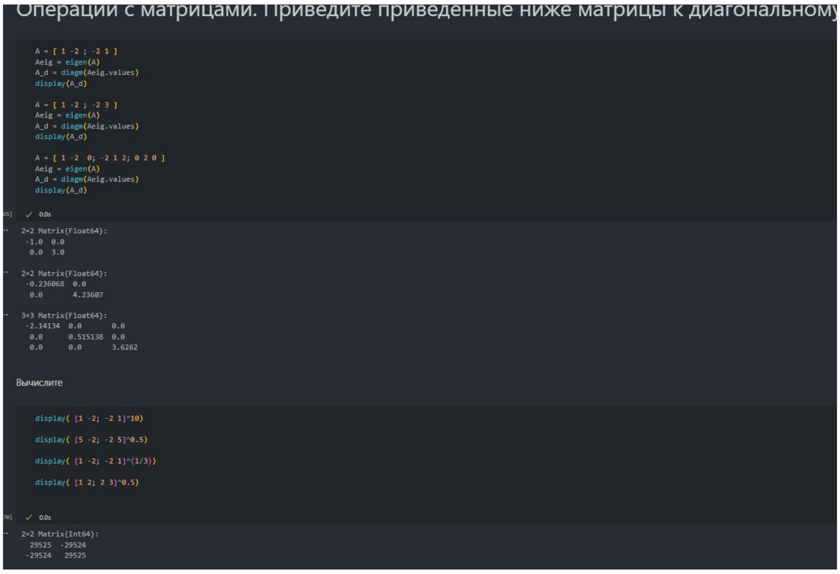



Рис. 4.15: Операции с матрицами

Вычислим (рис. 4.16).

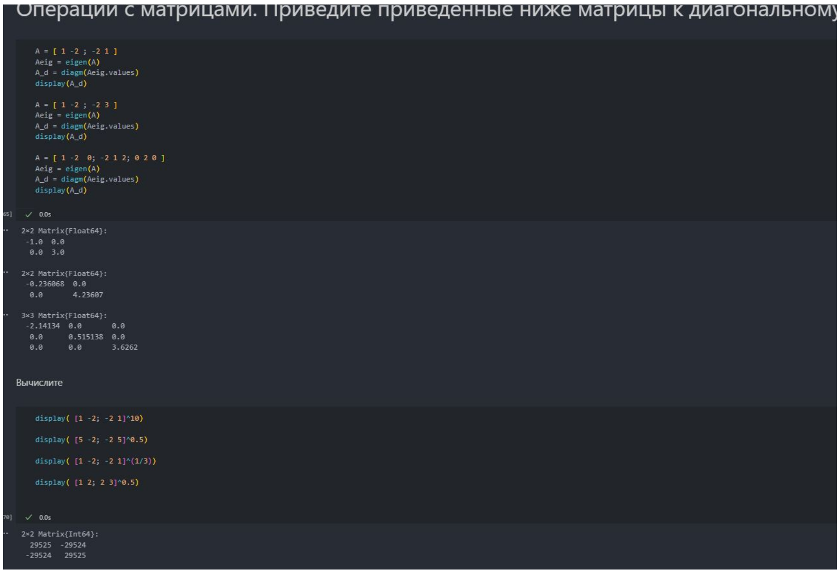
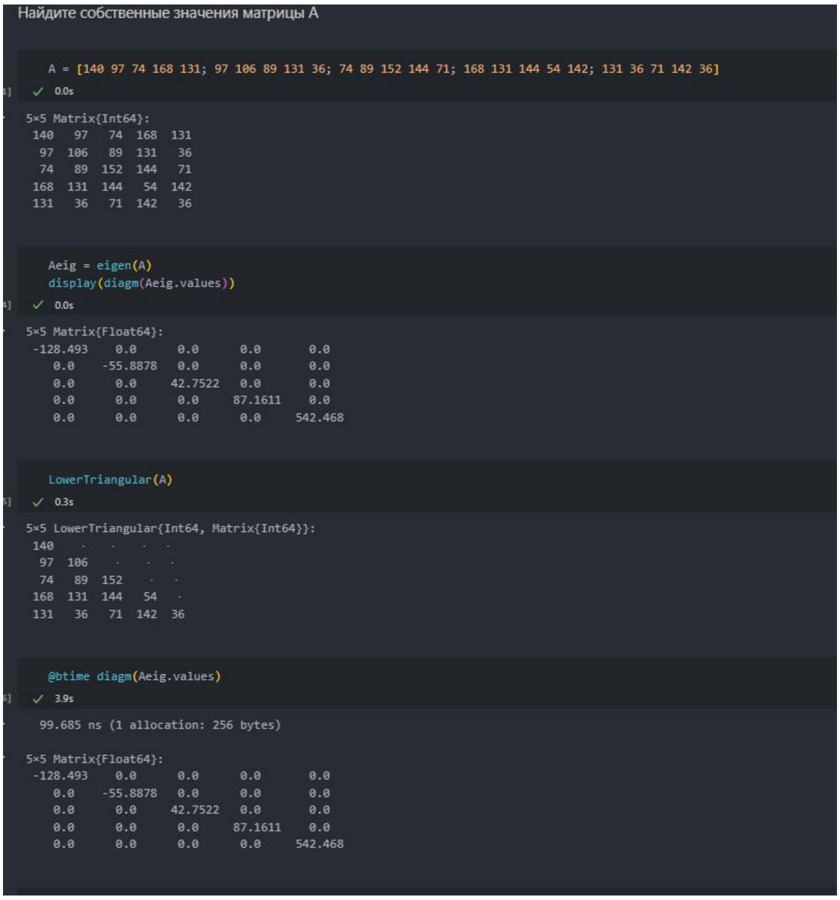



Рис. 4.16: Операции с матрицами

Найдем собственные значения матрицы A. Создадим диагональную матрицу из собственных значений матрицы A. Создадим нижнедиагональную матрицу из матрицы A. Оценим эффективность выполняемых операций (рис. 4.17).


Найдите собственные значения матрицы 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]
✓ 0.0s

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
```

```
Aeig = eigen(A)
display(diagm(Aeig.values))
✓ 0.0s

5×5 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)
✓ 0.3s

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

```
@btime diagm(Aeig.values)
✓ 3.9s

  99.685 ns (1 allocation: 256 bytes)

5×5 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
```

Рис. 4.17: Операции с матрицами

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 матрицып родуктивными (рис. 4.18-4.19).

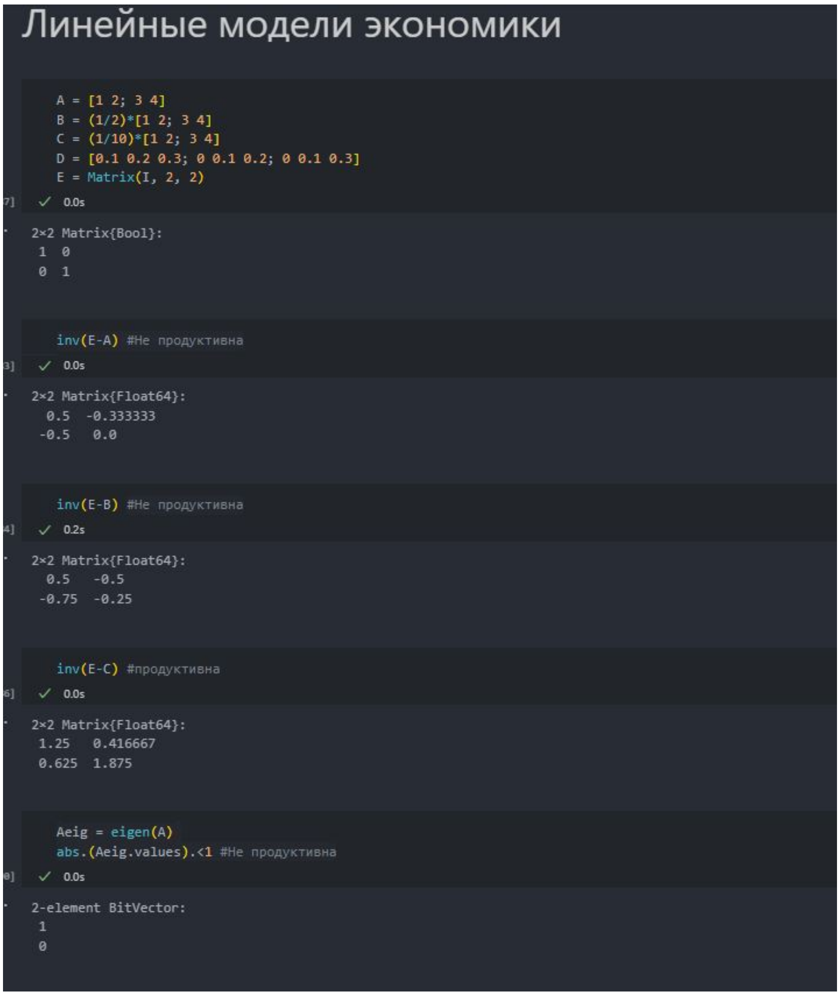



Рис. 4.18: Линейные модели экономики

```
Beig = eigen(B)
abs.(Beig.values).<1 #Не продуктивна
```

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

```
Ceig = eigen(C)
abs.(Ceig.values).<1 #продуктивна
```

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

```
Deig = eigen(D)
abs.(Deig.values).<1 #продуктивна
```

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

Рис. 4.19: Линейные модели экономики

# 5 Выводы

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

# 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).