Лабораторная работа №4.

Линейная алгебра.

Ишанова А.И. группа НФИ-02-19

Содержание

# Цель работы

Изучение возможностей специализированных пакетов Julia для выполнения и оценки эффективности операций над объектами линейной алгебры.

# Задание

1. Используя Jupyter Lab, повторите примеры из раздела 4.2.
2. Выполните задания для самостоятельной работы (раздел 4.4). [1]

# Выполнение лабораторной работы

## Повторение примеров

1. Повторяем примеры с поэлементными операциями над многомерными массивами. (fig. 1 - fig. 2)

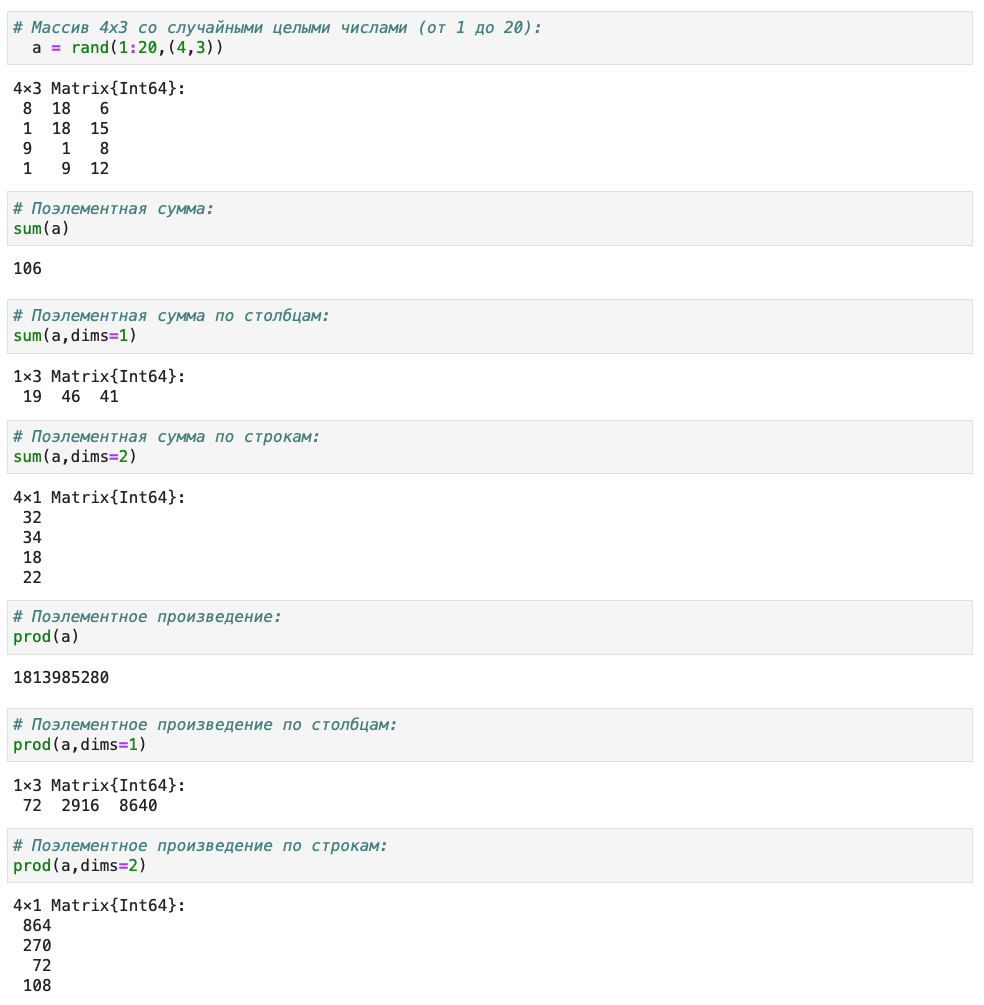


Figure 1: Примеры с поэлементными операциями над многомерными массивами - 1

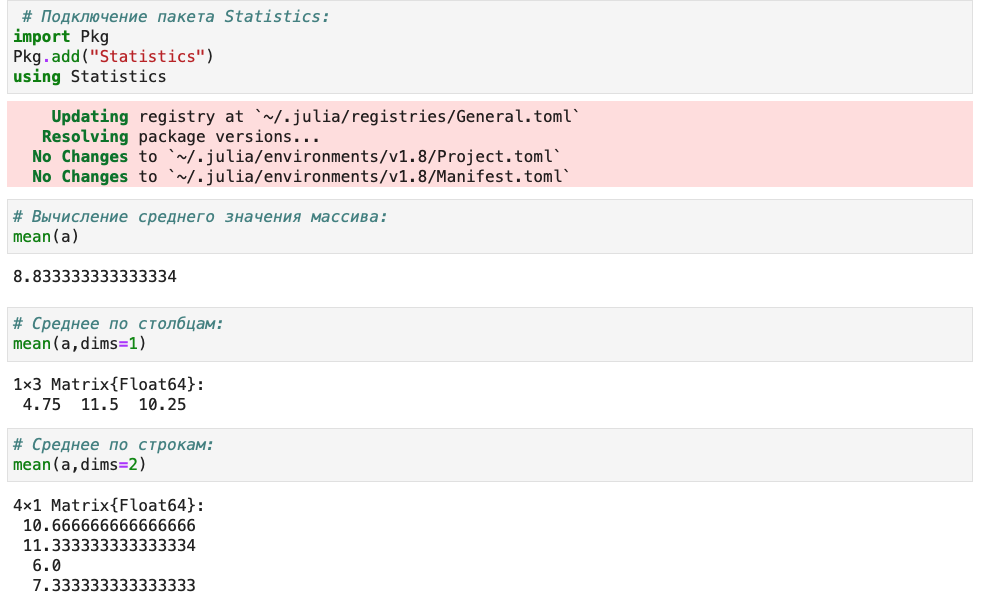


Figure 2: Примеры с поэлементными операциями над многомерными массивами - 2

1. Повторяем примеры с операциями над матрицами с пакетом LinearAlgebra (fig. 3 - fig. 4)

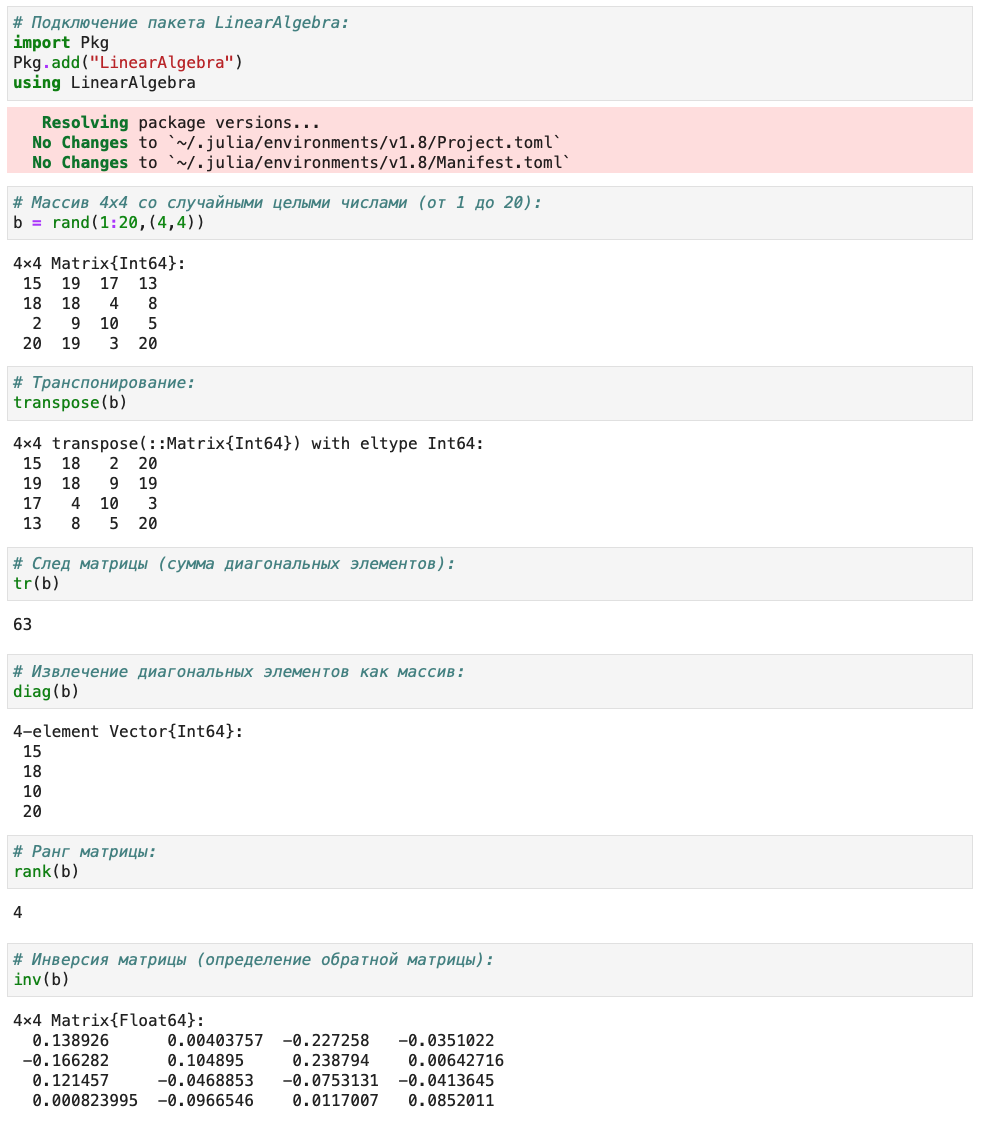


Figure 3: Примеры с операциями над матрицами с пакетом LinearAlgebra - 1

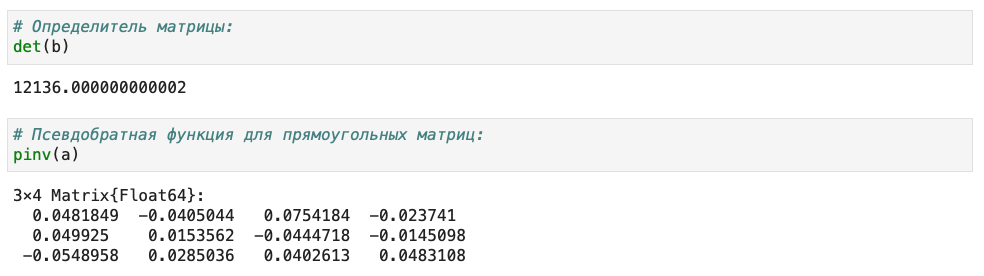


Figure 4: Примеры с операциями над матрицами с пакетом LinearAlgebra - 2

1. Повторяем примеры из раздела “Вычисление нормы векторов и матриц, повороты, вращения”. (fig. 5 - fig. 7)

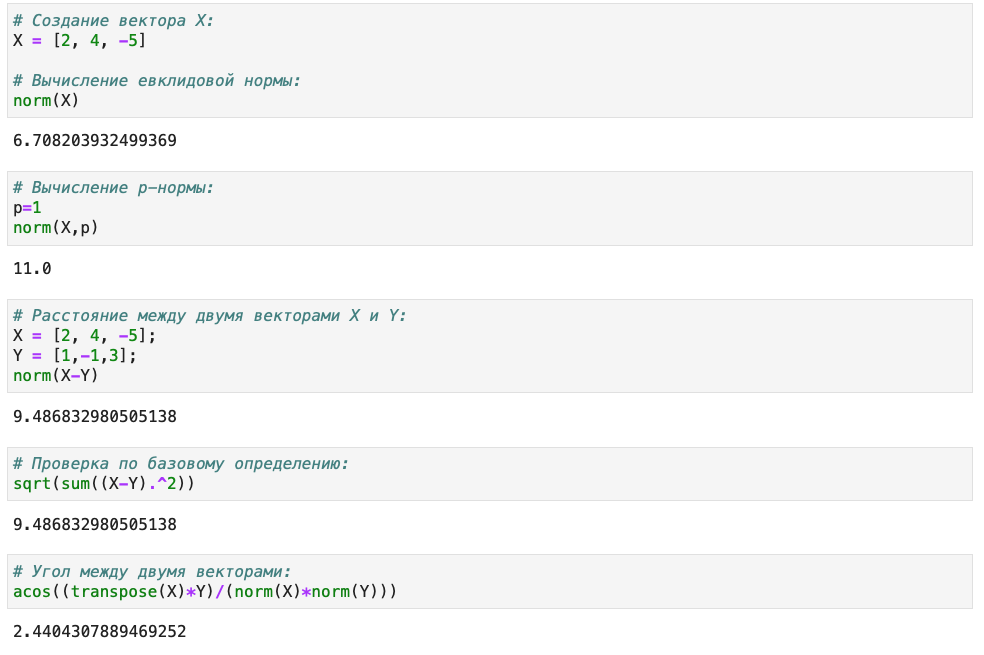


Figure 5: Примеры из раздела “Вычисление нормы векторов и матриц, повороты, вращения” - 1

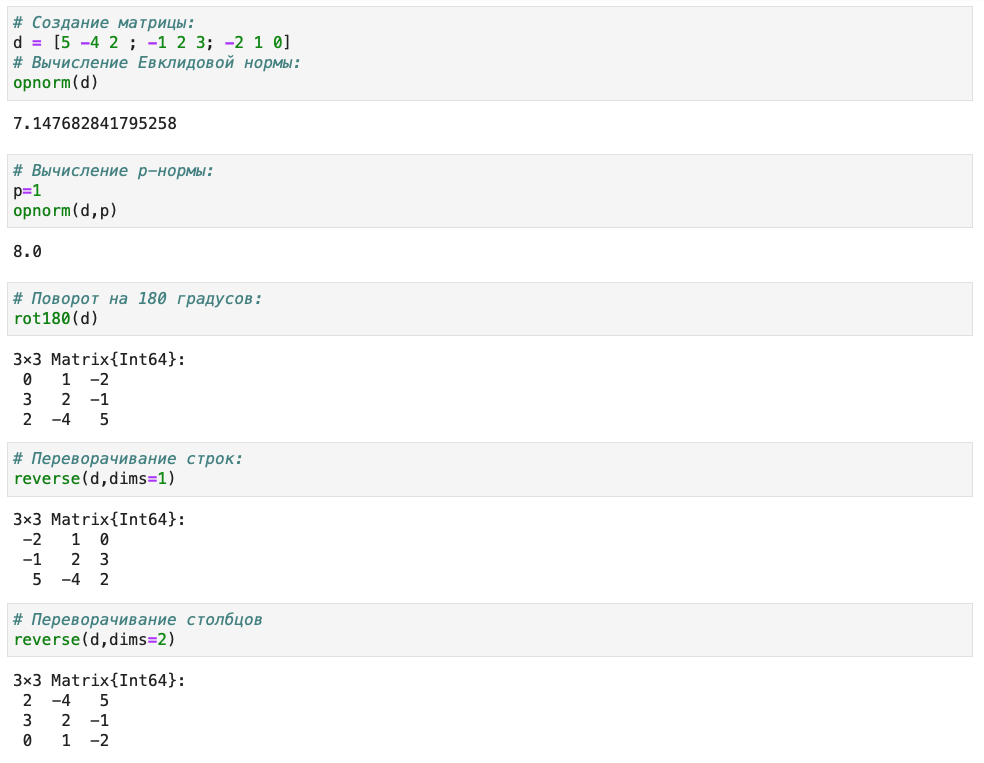


Figure 6: Примеры из раздела “Вычисление нормы векторов и матриц, повороты, вращения” - 2

1. Повторяем примеры из раздела “Матричное умножение, единичная матрица, скалярное произведение”. (fig. 7)

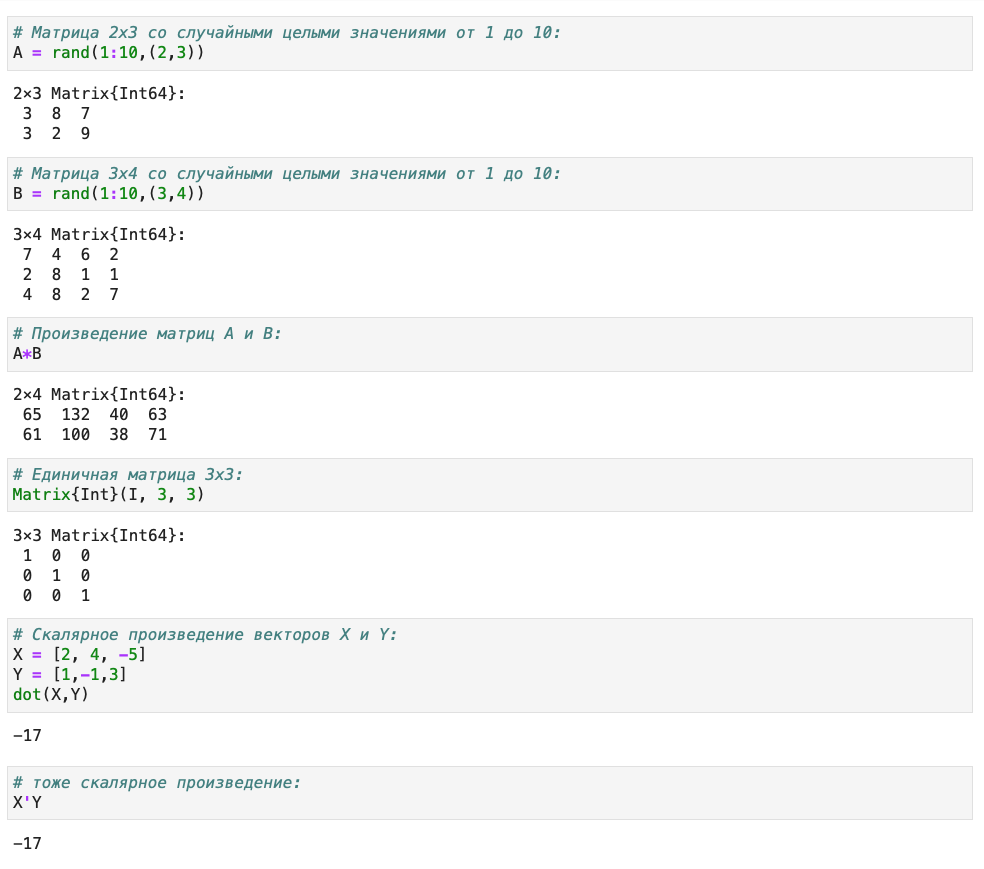


Figure 7: Примеры из раздела “Матричное умножение, единичная матрица, скалярное произведение”

1. Повторяем примеры из раздела “Факторизация. Специальные матричные структуры”. (fig. 8 - fig. 17)

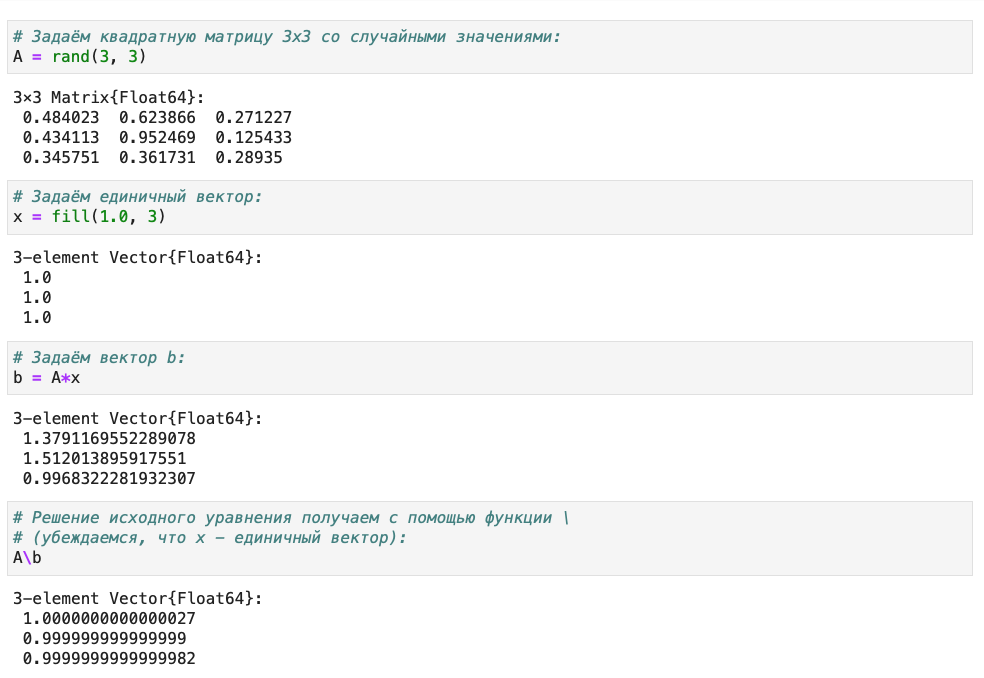


Figure 8: Примеры из раздела “Факторизация. Специальные матричные структуры” - 1

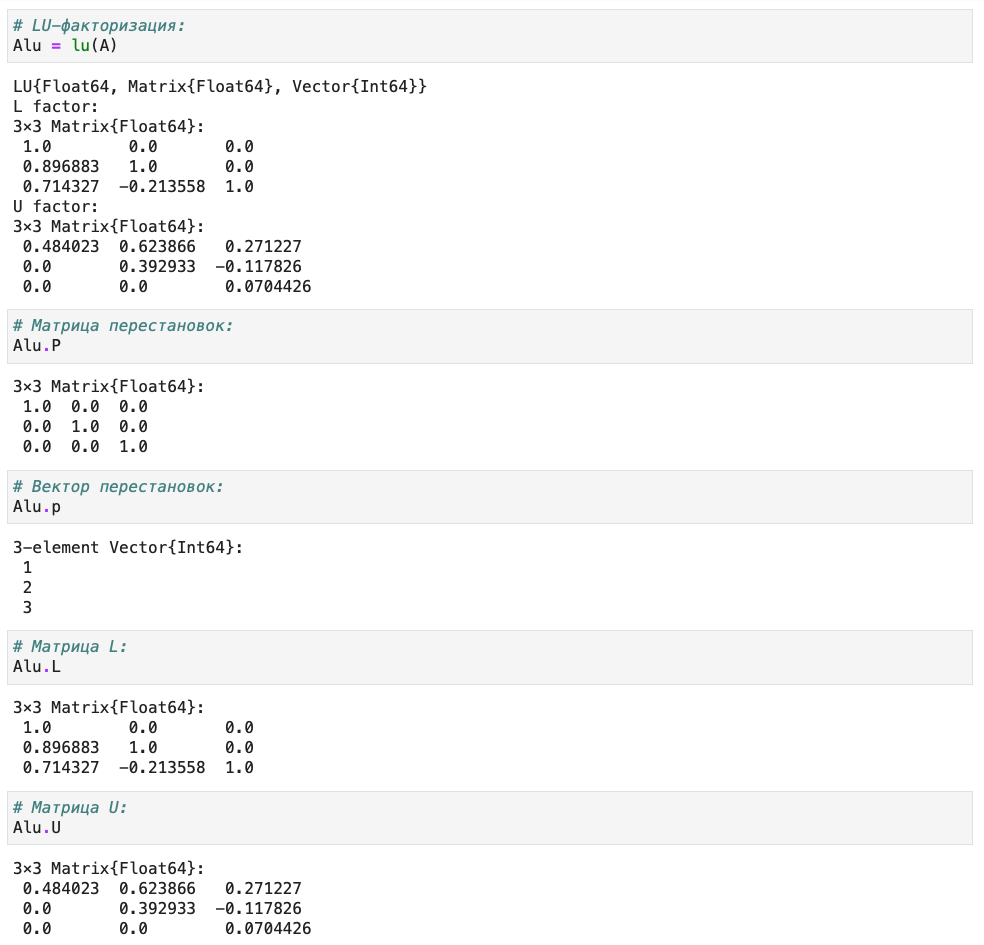


Figure 9: Примеры из раздела “Факторизация. Специальные матричные структуры” - 2

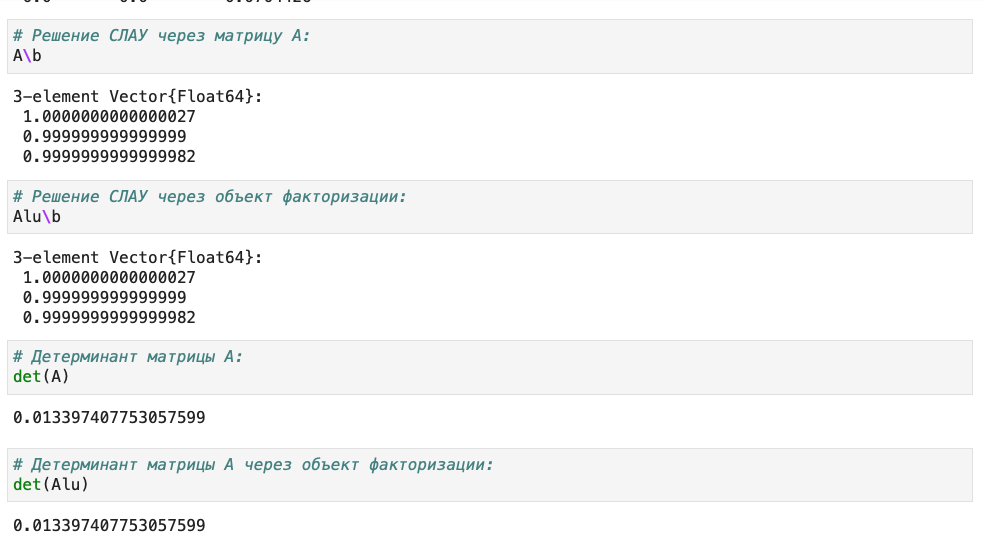


Figure 10: Примеры из раздела “Факторизация. Специальные матричные структуры” - 3

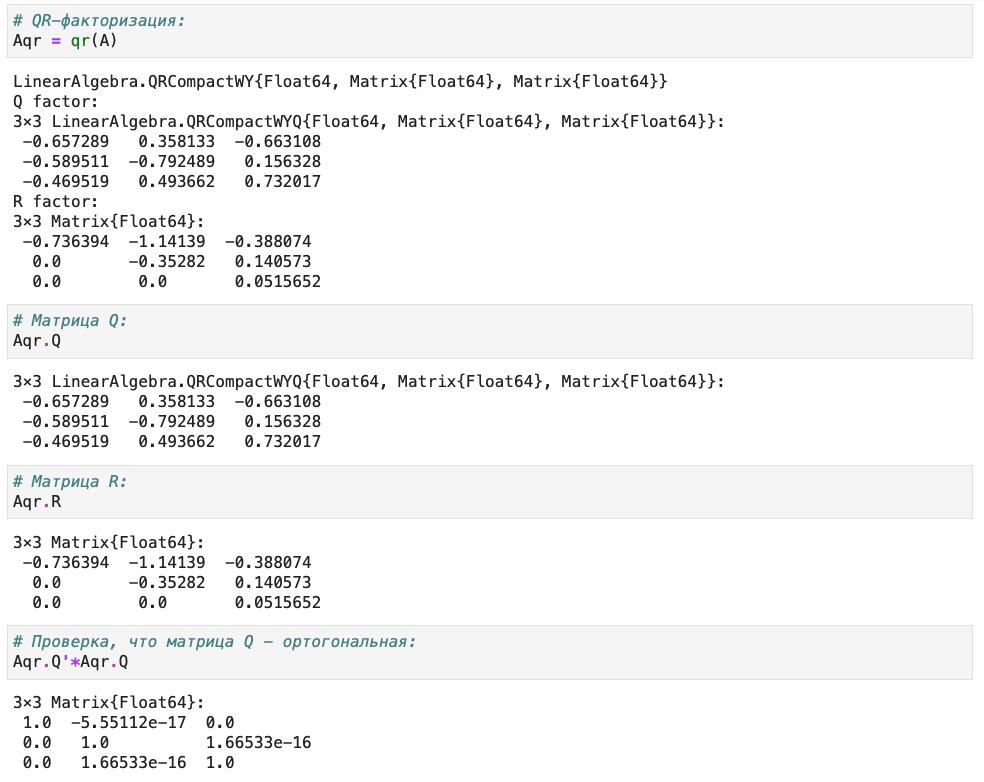


Figure 11: Примеры из раздела “Факторизация. Специальные матричные структуры” - 4

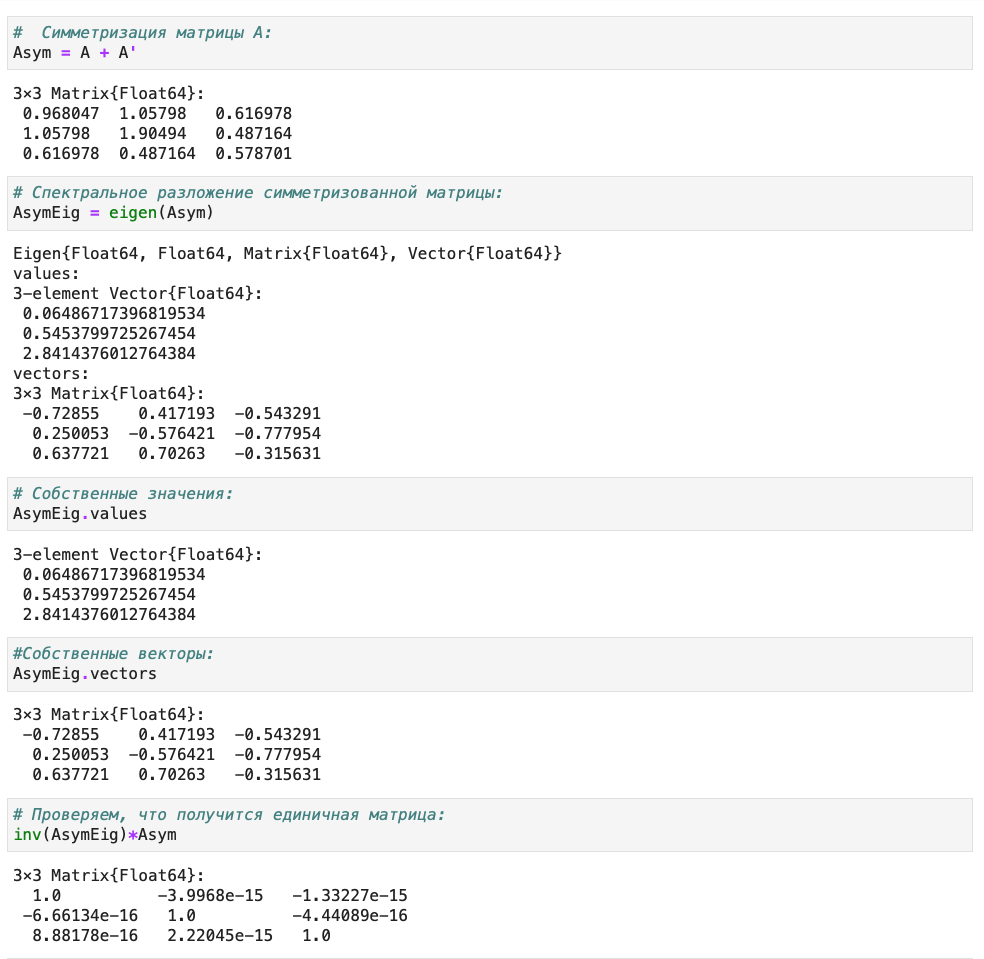


Figure 12: Примеры из раздела “Факторизация. Специальные матричные структуры” - 5

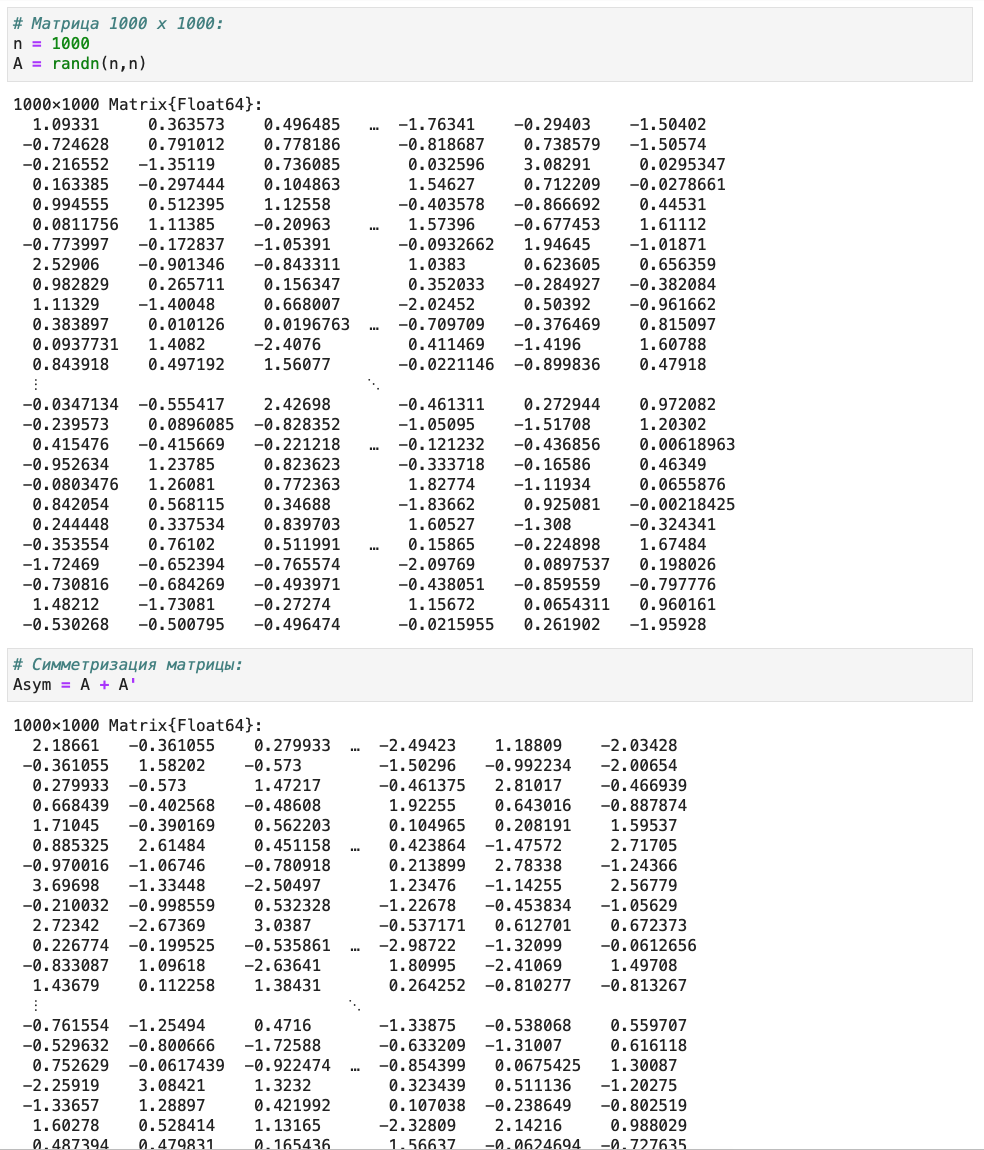


Figure 13: Примеры из раздела “Факторизация. Специальные матричные структуры” - 6

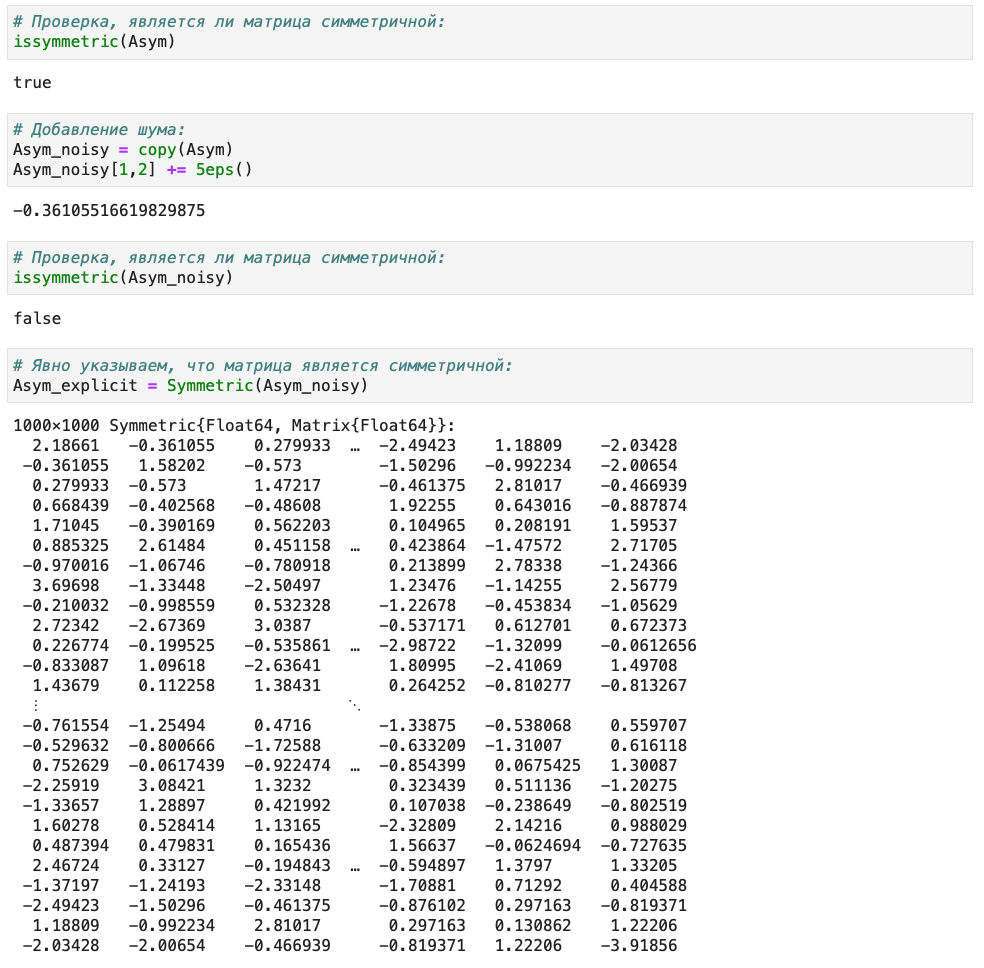


Figure 14: Примеры из раздела “Факторизация. Специальные матричные структуры” - 7

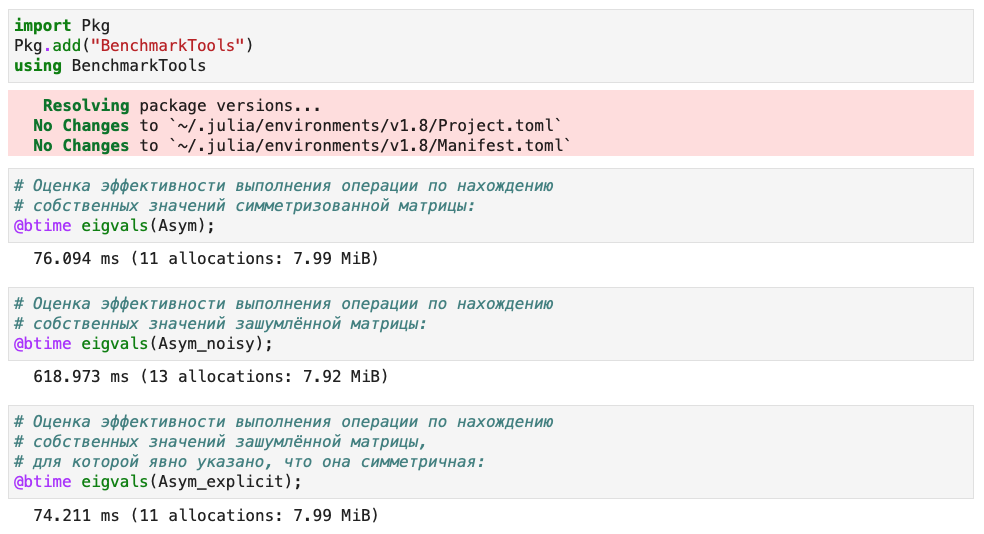


Figure 15: Примеры из раздела “Факторизация. Специальные матричные структуры” - 8

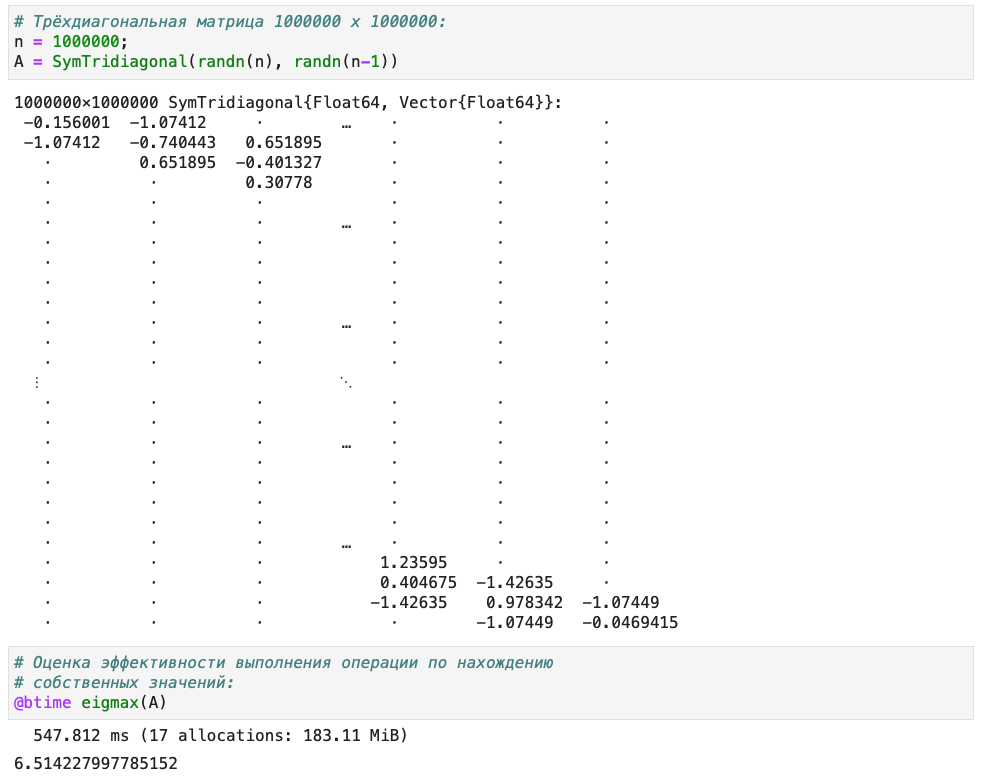


Figure 16: Примеры из раздела “Факторизация. Специальные матричные структуры” - 9

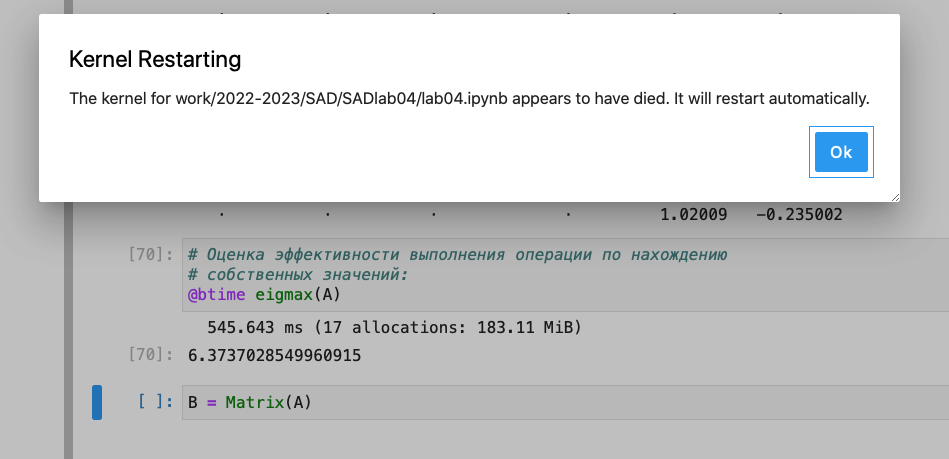


Figure 17: Ошибка при выполнении команды B = Matrix(A)

1. Повторяем примеры из раздела “Общая линейная алгебра”. (fig. 18)

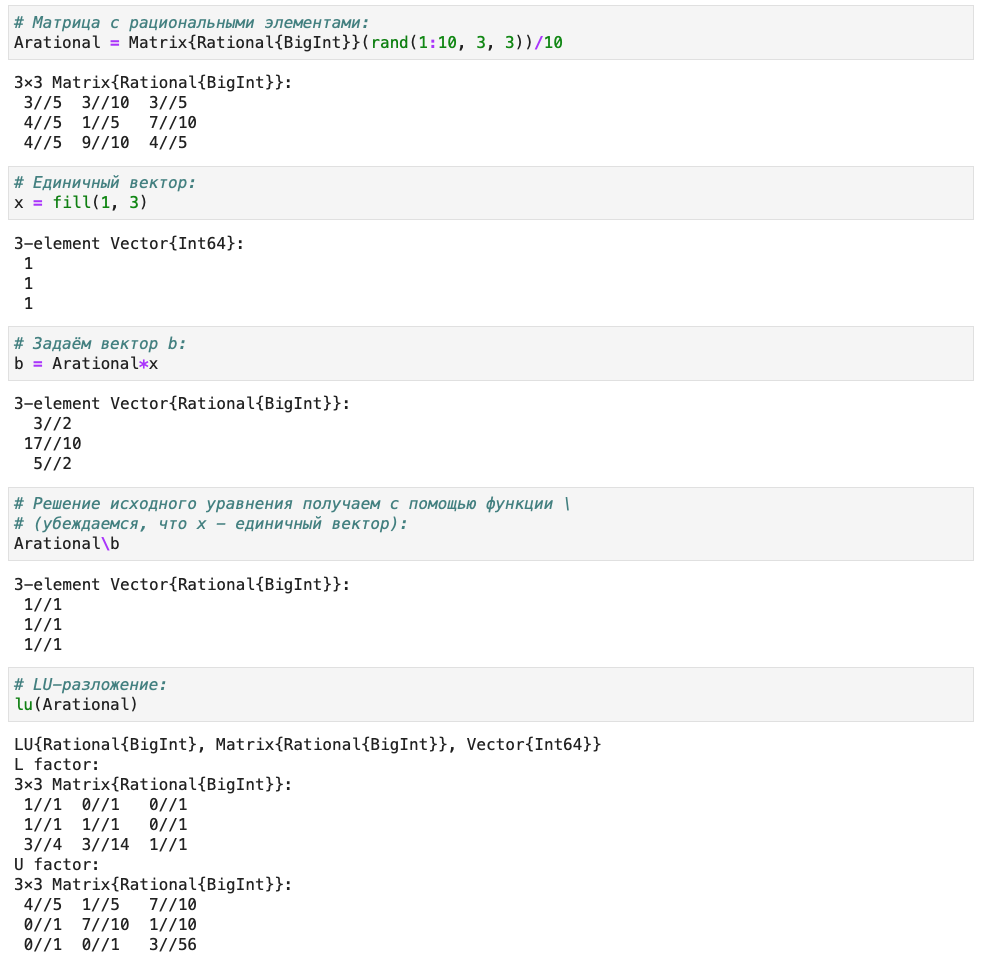


Figure 18: Примеры из раздела “Общая линейная алгебра”

## Задания для самостоятельной работы

1. Призведение векторов.

1.1. Задайте вектор v. Умножьте вектор v скалярно сам на себя и сохраните результат в dot\_v. (fig. 19)

1.2. Умножьте v матрично на себя (внешнее произведение), присвоив результат переменной outer\_v. (fig. 19)

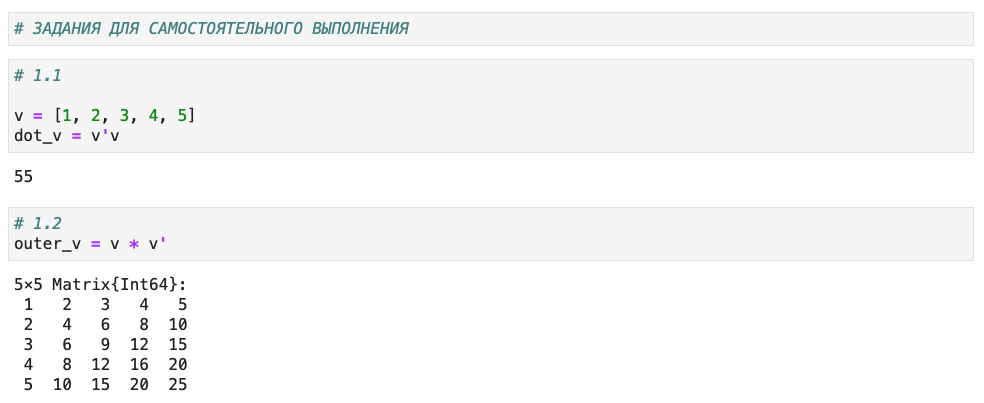


Figure 19: Скалярное и матричное умножение вектора на себя

1. Системы линейных уравнений.

2.1. Решить СЛАУ с двумя неизвестными.

1. (fig. 20)
2. Система имеет бесконечно много решений, поэтому программа выдает ошибку. (fig. 20)
3. Система не имеет решений, поэтому программа выдает ошибку. (fig. 20)
4. (fig. 21)
5. (fig. 21)
6. (fig. 21)



Figure 20: Решение СЛАУ с двумя неизвестными - 1

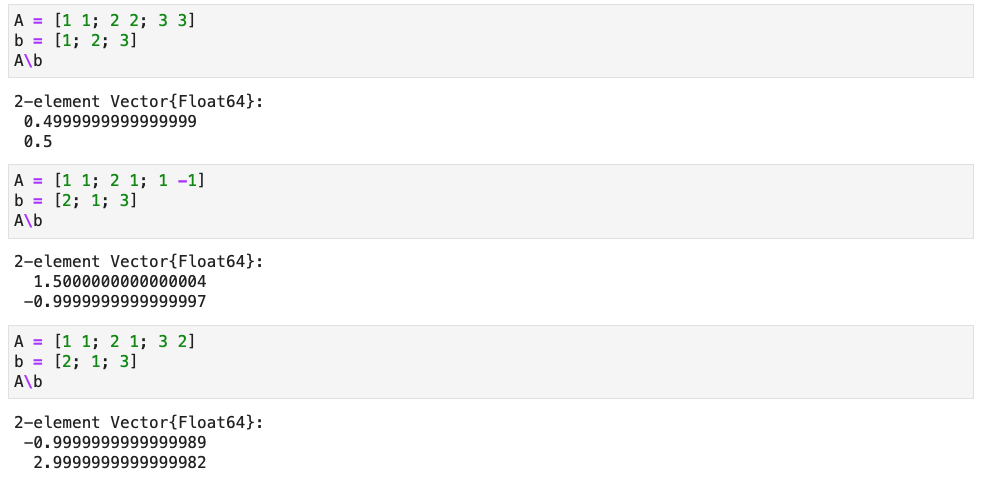


Figure 21: Решение СЛАУ с двумя неизвестными - 2

2.2. Решить СЛАУ с тремя неизвестными.

1. (fig. 22)
2. (fig. 22)
3. Система имеет бесконечно много решений, поэтому программа выдает ошибку.(fig. 22)
4. Система не имеет решений, поэтому программа выдает ошибку. (fig. 23)

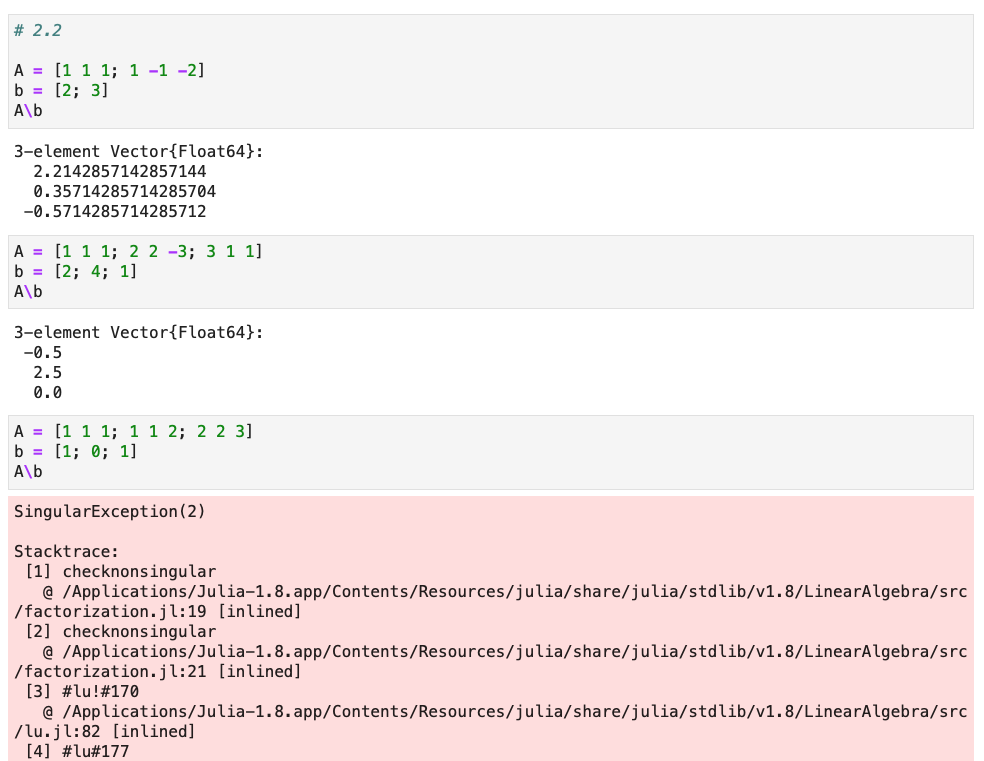


Figure 22: Решение СЛАУ с тремя неизвестными - 1



Figure 23: Решение СЛАУ с тремя неизвестными - 2

1. Операции с матрицами.

3.1. Приведите матрицы к диагональному виду.

1. (fig. 24)
2. (fig. 24)
3. (fig. 24)

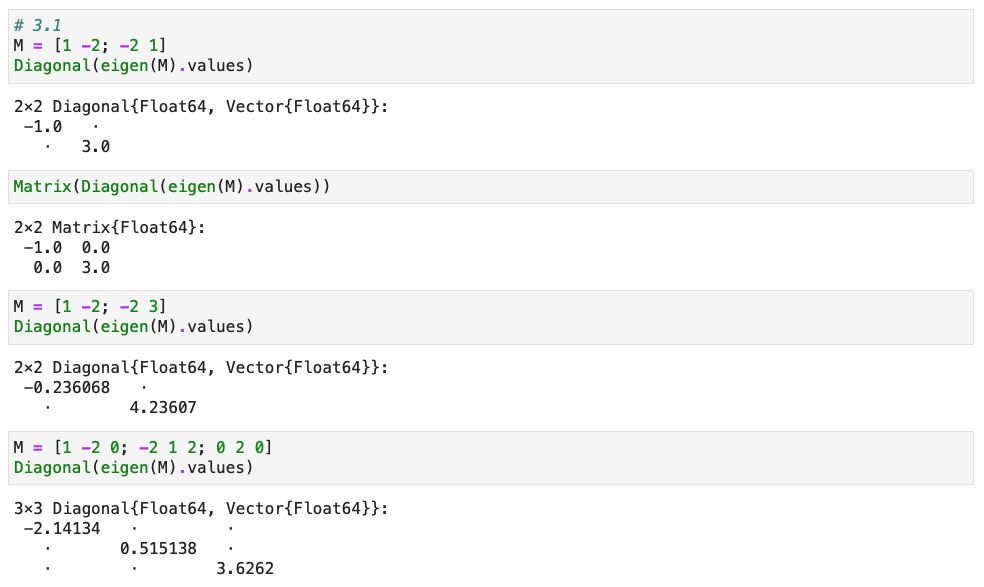


Figure 24: Приведение матриц к диагональному виду

3.2. Вычислить:

1. (fig. 25)
2. (fig. 25)
3. (fig. 25)
4. (fig. 25)

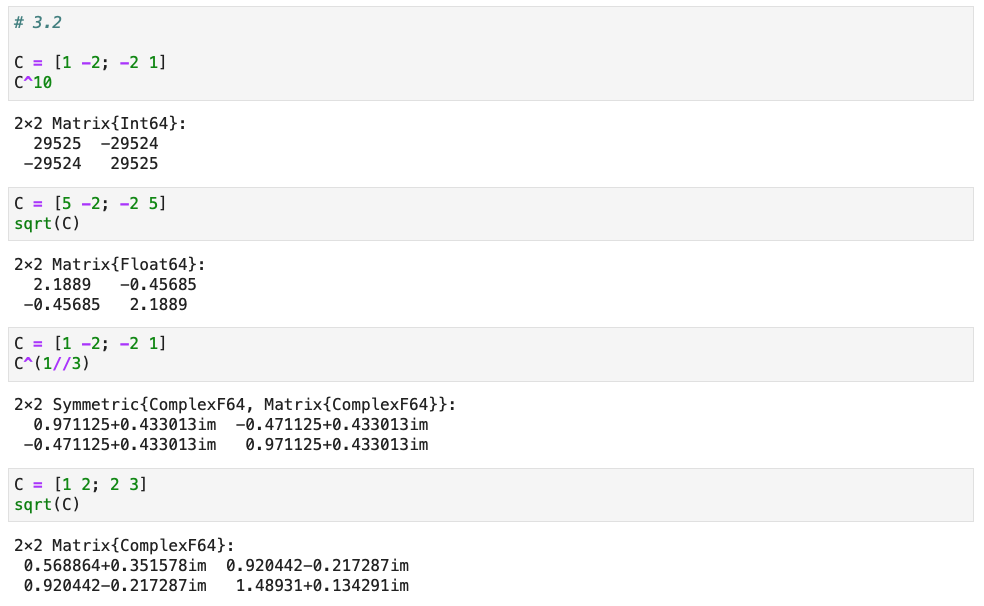


Figure 25: Вычисление операций над матрицами

3.3. Найти собственные значения, создать диагональную матрицу из собственных значений и нижнедиагональную матрицу из исходной, оценить эффективность выполнения операций. (fig. 26)

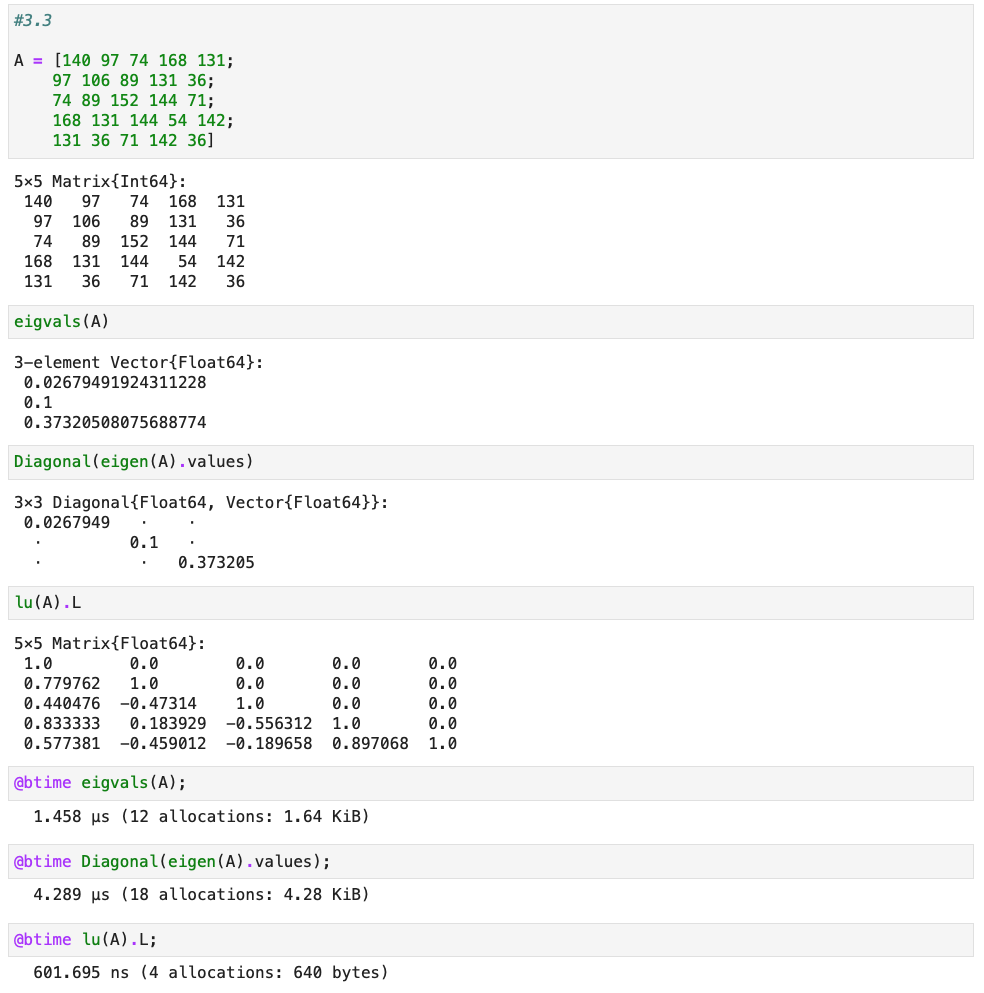


Figure 26: Нахождение собственных значений, составление диагональной матрицы из собственных значений, составление нижнедтиагонально матрицы и оценивание эффективности этих операций.

1. Линейные модели экономики.

Линейная модель:

Проверить являются ли матрицы A продуктивными по критерию:

4.1. Критерий: Решение системы х при любом неотрицательном y имеет только неотрицательные элементы.

1. (fig. 27)

b)(fig. 27)

c)(fig. 27)

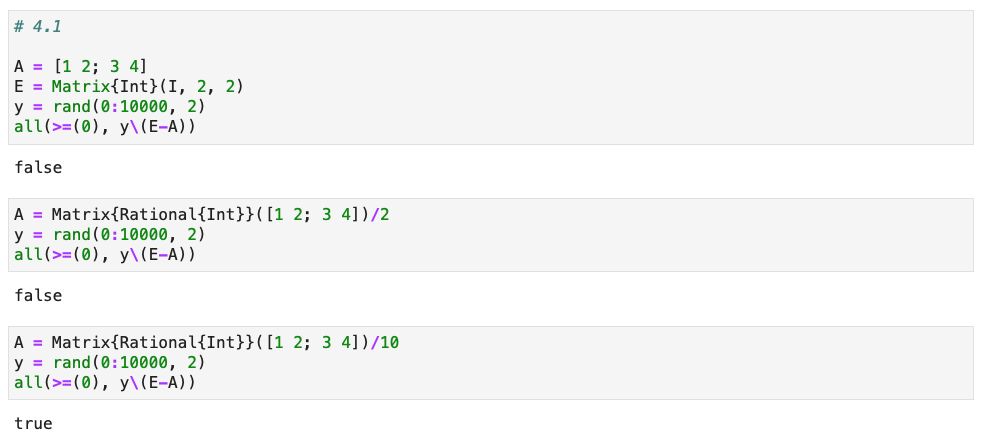


Figure 27: Проверка продуктивности матрицы А по критерию 1

4.2. Все элементы матрицы - неотрицательные числа.

1. (fig. 28)

b)(fig. 28)

c)(fig. 28)

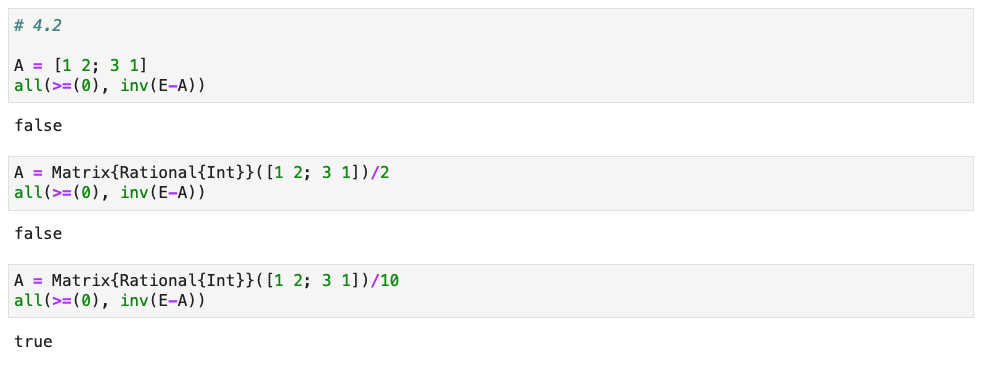


Figure 28: Проверка продуктивности матрицы А по критерию 2

4.3. Все собственные значения матрицы А по модулю меньше 1.

1. (fig. 29)

b)(fig. 29)

c)(fig. 29)

d)(fig. 29)

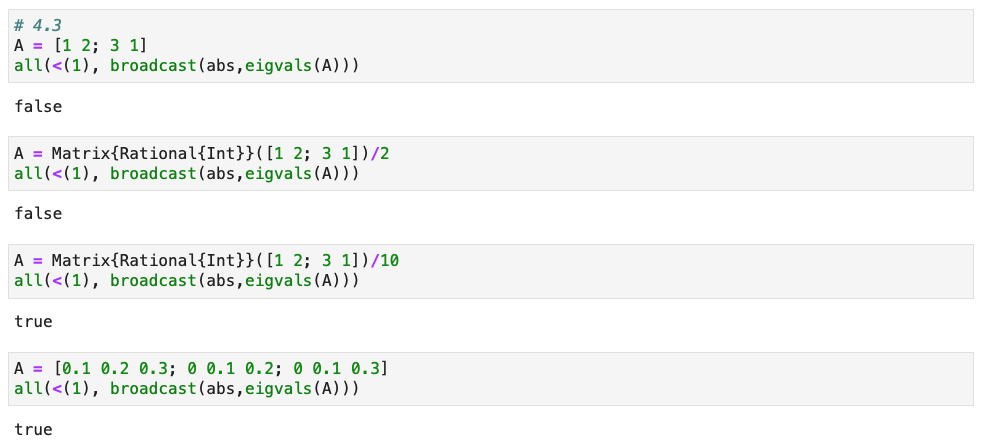


Figure 29: Проверка продуктивности матрицы А по критерию 3

# Листинг

# -\*- coding: utf-8 -\*-  
# ---  
# jupyter:  
# jupytext:  
# text\_representation:  
# extension: .jl  
# format\_name: light  
# format\_version: '1.5'  
# jupytext\_version: 1.14.1  
# kernelspec:  
# display\_name: Julia 1.8.2  
# language: julia  
# name: julia-1.8  
# ---  
  
# Массив 4x3 со случайными целыми числами (от 1 до 20):  
 a = rand(1:20,(4,3))  
  
# 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)  
  
 # Подключение пакета Statistics:  
import Pkg  
Pkg.add("Statistics")  
using Statistics  
  
# Вычисление среднего значения массива:  
mean(a)  
  
# Среднее по столбцам:  
mean(a,dims=1)  
  
# Среднее по строкам:  
mean(a,dims=2)  
  
# Подключение пакета LinearAlgebra:  
import Pkg  
Pkg.add("LinearAlgebra")  
using LinearAlgebra  
  
# Массив 4x4 со случайными целыми числами (от 1 до 20):  
b = rand(1:20,(4,4))  
  
# Транспонирование:  
transpose(b)  
  
# След матрицы (сумма диагональных элементов):  
tr(b)  
  
# Извлечение диагональных элементов как массив:  
diag(b)  
  
# Ранг матрицы:  
rank(b)  
  
# Инверсия матрицы (определение обратной матрицы):  
inv(b)  
  
# Определитель матрицы:  
det(b)  
  
# Псевдобратная функция для прямоугольных матриц:  
pinv(a)  
  
# +  
# 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))  
  
# 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:  
opnorm(d)  
  
# Вычисление p-нормы:  
p=1  
opnorm(d,p)  
  
# Поворот на 180 градусов:  
rot180(d)  
  
# Переворачивание строк:  
reverse(d,dims=1)  
  
# Переворачивание столбцов  
reverse(d,dims=2)  
  
# Матрица 2x3 со случайными целыми значениями от 1 до 10:  
A = rand(1:10,(2,3))  
  
# 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  
  
# 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  
  
# LU-факторизация:  
Alu = lu(A)  
  
# Матрица перестановок:  
Alu.P  
  
# Вектор перестановок:  
Alu.p  
  
# Матрица L:  
Alu.L  
  
# Матрица U:  
Alu.U  
  
# Решение СЛАУ через матрицу A:  
A\b  
  
# Решение СЛАУ через объект факторизации:  
Alu\b  
  
# Детерминант матрицы A:  
det(A)  
  
# Детерминант матрицы A через объект факторизации:  
det(Alu)  
  
# QR-факторизация:  
Aqr = qr(A)  
  
# Матрица Q:  
Aqr.Q  
  
# Матрица R:  
Aqr.R  
  
# Проверка, что матрица Q - ортогональная:  
Aqr.Q'\*Aqr.Q  
  
# Симметризация матрицы A:  
Asym = A + A'  
  
# Спектральное разложение симметризованной матрицы:  
AsymEig = eigen(Asym)  
  
# Собственные значения:  
AsymEig.values  
  
#Собственные векторы:  
AsymEig.vectors  
  
# Проверяем, что получится единичная матрица:  
inv(AsymEig)\*Asym  
  
# Матрица 1000 х 1000:  
n = 1000  
A = randn(n,n)  
  
# Симметризация матрицы:  
Asym = A + A'  
  
# Проверка, является ли матрица симметричной:  
issymmetric(Asym)  
  
# Добавление шума:  
Asym\_noisy = copy(Asym)  
Asym\_noisy[1,2] += 5eps()  
  
# Проверка, является ли матрица симметричной:  
issymmetric(Asym\_noisy)  
  
# Явно указываем, что матрица является симметричной:  
Asym\_explicit = Symmetric(Asym\_noisy)  
  
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);  
  
# Оценка эффективности выполнения операции по нахождению  
# собственных значений зашумлённой матрицы,  
# для которой явно указано, что она симметричная:  
@btime eigvals(Asym\_explicit);  
  
# Трёхдиагональная матрица 1000000 х 1000000:  
n = 1000000;  
A = SymTridiagonal(randn(n), randn(n-1))  
  
# + tags=[]  
# Оценка эффективности выполнения операции по нахождению  
# собственных значений:  
@btime eigmax(A)  
# -  
  
B = Matrix(A)  
  
# Матрица с рациональными элементами:  
Arational = Matrix{Rational{BigInt}}(rand(1:10, 3, 3))/10  
  
# Единичный вектор:  
x = fill(1, 3)  
  
# Задаём вектор b:  
b = Arational\*x  
  
# Решение исходного уравнения получаем с помощью функции \   
# (убеждаемся, что x - единичный вектор):  
Arational\b  
  
# LU-разложение:  
lu(Arational)  
  
# +  
# ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОГО ВЫПОЛНЕНИЯ  
  
# +  
# 1.1  
  
v = [1, 2, 3, 4, 5]  
dot\_v = v'v  
# -  
  
# 1.2  
outer\_v = v \* v'  
  
# +  
#2.1  
  
A = [ 1 1; 1 -1]  
b = [2;3]  
A\b  
# -  
  
A = [1 1; 2 2]  
b = [4;2]  
A\b  
  
A = [1 1; 2 2]  
b = [2;5]  
A\b  
  
A = [1 1; 2 2; 3 3]  
b = [1; 2; 3]  
A\b  
  
A = [1 1; 2 1; 1 -1]  
b = [2; 1; 3]  
A\b  
  
A = [1 1; 2 1; 3 2]  
b = [2; 1; 3]  
A\b  
  
# +  
# 2.2  
  
A = [1 1 1; 1 -1 -2]  
b = [2; 3]  
A\b  
# -  
  
A = [1 1 1; 2 2 -3; 3 1 1]  
b = [2; 4; 1]  
A\b  
  
A = [1 1 1; 1 1 2; 2 2 3]  
b = [1; 0; 1]  
A\b  
  
A = [1 1 1; 1 1 2; 2 2 3]  
b = [1; 0; 0]  
A\b  
  
# 3.1  
M = [1 -2; -2 1]  
Diagonal(eigen(M).values)  
  
Matrix(Diagonal(eigen(M).values))  
  
M = [1 -2; -2 3]  
Diagonal(eigen(M).values)  
  
M = [1 -2 0; -2 1 2; 0 2 0]  
Diagonal(eigen(M).values)  
  
# +  
# 3.2  
  
C = [1 -2; -2 1]  
C^10  
# -  
  
C = [5 -2; -2 5]  
sqrt(C)  
  
C = [1 -2; -2 1]  
C^(1//3)  
  
C = [1 2; 2 3]  
sqrt(C)  
  
# +  
#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]  
# -  
  
eigvals(A)  
  
Diagonal(eigen(A).values)  
  
lu(A).L  
  
@btime eigvals(A);  
  
@btime Diagonal(eigen(A).values);  
  
@btime lu(A).L;  
  
# +  
# 4.1  
  
A = [1 2; 3 4]  
E = Matrix{Int}(I, 2, 2)  
y = rand(0:10000, 2)  
all(>=(0), y\(E-A))  
# -  
  
A = Matrix{Rational{Int}}([1 2; 3 4])/2  
y = rand(0:10000, 2)  
all(>=(0), y\(E-A))  
  
A = Matrix{Rational{Int}}([1 2; 3 4])/10  
y = rand(0:10000, 2)  
all(>=(0), y\(E-A))  
  
# +  
# 4.2  
  
A = [1 2; 3 1]  
all(>=(0), inv(E-A))  
# -  
  
A = Matrix{Rational{Int}}([1 2; 3 1])/2  
all(>=(0), inv(E-A))  
  
A = Matrix{Rational{Int}}([1 2; 3 1])/10  
all(>=(0), inv(E-A))  
  
# 4.3  
A = [1 2; 3 1]  
all(<(1), broadcast(abs,eigvals(A)))  
  
A = Matrix{Rational{Int}}([1 2; 3 1])/2  
all(<(1), broadcast(abs,eigvals(A)))  
  
A = Matrix{Rational{Int}}([1 2; 3 1])/10  
all(<(1), broadcast(abs,eigvals(A)))  
  
A = [0.1 0.2 0.3; 0 0.1 0.2; 0 0.1 0.3]  
all(<(1), broadcast(abs,eigvals(A)))

# Вывод

В ходе выполнения лабораторной работы операции применимые в рамках задач линейной алгебры. Были изучены возможности специализированных пакетов Julia для выполнения и оценки эффективности операций над объектами линейной алгебры. Также были поставлены пакеты LinearAlgebra, Statistics и BenchmarkTools.

# Библиография

1. Методические материалы курса.