Российский университет дружбы народов

Факультет физико-математических и естественных наук

Отчёт по лабораторной работе №4

1032203967
Быстров Глеб

Москва 2023

Цель работы (задание)

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

Задачи (метод выполнения)

- Поэлементные операции над многомерными массивами

```
# Массив 4x3 со случайными целыми числами (от 1 до 20):
a = rand(1:20,(4,3))
```

```
4x3 Matrix{Int64}:
  9   9  18
  6   2  19
  8  12   8
 20   2  11
```

```
# Поэлементная сумма:
sum(a)
```

```
124
```

```
# Поэлементная сумма по столбцам:
sum(a,dims=1)
```

```
1x3 Matrix{Int64}:
 43  25  56
```

```
# Поэлементная сумма по строкам:
sum(a,dims=2)
```

```
4x1 Matrix{Int64}:
 36
 27
 28
 33
```

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

```
    Updating registry at `C:\Users\GlebB\.julia\registries\General.toml`
   Resolving package versions...
    Updating `C:\Users\GlebB\.julia\environments\v1.9\Project.toml`
  [10745b16] + Statistics v1.9.0
  No Changes to `C:\Users\GlebB\.julia\environments\v1.9\Manifest.toml`
```

```
4x1 Matrix{Float64}:
 12.0
  9.0
  9.333333333333334
 11.0
```

```
# Вычисление среднего значения массива:
mean(a)
```

```
10.333333333333334
```

```
# Среднее по столбцам:
mean(a,dims=1)
```

```
1x3 Matrix{Float64}:
 10.75  6.25  14.0
```

```
# Среднее по строкам:
mean(a,dims=2)
```

```
4x1 Matrix{Float64}:
 12.0
  9.0
```

Задачи (метод выполнения)

- Транспонирование, след, ранг, определитель и инверсия матрицы

```
# Подключение пакета LinearAlgebra:
import Pkg
Pkg.add("LinearAlgebra")
using LinearAlgebra
```

```
   Resolving package versions...
    Updating `C:\Users\GlebB\.julia\environments\v1.9\Project.toml`
  [37e2e46d] + LinearAlgebra
  No Changes to `C:\Users\GlebB\.julia\environments\v1.9\Manifest.toml`
```

```
# Массив 4x4 со случайными целыми числами (от 1 до 20):
b = rand(1:20,(4,4))
# Транспонирование:
transpose(b)
```

```
4×4 transpose(::Matrix{Int64}) with eltype Int64:
  1  11   6   6
 12   5  14   7
  6  11   5  13
  2  12  14  15
```

```
# След матрицы (сумма диагональных элементов):
tr(b)
```

```
26
```

```
# Извлечение диагональных элементов как массив:
diag(b)
```

```
4-element Vector{Int64}:
  8
 12
 18
 15
```

```
# Ранг матрицы:
rank(b)
```

```
4
```

```
# Инверсия матрицы (определение обратной матрицы):
inv(b)
```

```
4×4 Matrix{Float64}:
 -0.000289603   0.16044      0.0133217  -0.140747
  0.0643885    -0.00453712   0.0381311  -0.0405445
  0.067381      0.00424751  -0.099527    0.0805097
 -0.088329     -0.0657399    0.0631335   0.0721112
```

```
# Определитель матрицы:
det(b)
```

```
10359.0
```

```
# Псевдобратная функция для прямоугольных матриц:
pinv(a)
```

```
3×4 Matrix{Float64}:
 -0.014341   -0.0241341   0.00445983   0.0619098
  0.023717   -0.0373237   0.0767987   -0.030195
  0.0212517   0.0510161  -0.0217465   -0.0161695
```

Задачи (метод выполнения)

- Вычисление нормы векторов и матриц, повороты, вращения

```
# Создание вектора X:
X = [2, 4, -5]
# Вычисление евклидовой нормы:
norm(X)
```

```
6.708203932499369
```

```
# Вычисление p-нормы:
p = 1
norm(X,p)
```

```
11.0
```

```
# Расстояние между двумя векторами X и Y:
X = [2, 4, -5];
Y = [1,-1,3];
norm(X-Y)
```

```
9.486832980505138
```

```
# Проверка по базовому определению:
sqrt(sum((X-Y).^2))
```

```
9.486832980505138
```

```
# Угол между двумя векторами:
acos((transpose(X)*Y)/(norm(X)*norm(Y)))
```

```
2.4404307889469252
```

```
# Создание матрицы:
d = [5 -4 2 ; -1 2 3; -2 1 0]
```

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

```
# Вычисление Евклидовой нормы:
opnorm(d)
```

```
7.147682841795258
```

```
# Вычисление p-нормы:
p=1
opnorm(d,p)
```

```
8.0
```

```
# Поворот на 180 градусов:
rot180(d)
```

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

```
# Переворачивание строк:
reverse(d,dims=1)
```

```
3×3 Matrix{Int64}:
```

Задачи (метод выполнения)

- Матричное умножение, единичная матрица, скалярное произведение

```
# Матрица 2x3 со случайными целыми значениями от 1 до 10:
A = rand(1:10,(2,3))
```

```
2×3 Matrix{Int64}:
 6  9  10
 8  4  10
```

```
# Матрица 3x4 со случайными целыми значениями от 1 до 10:
B = rand(1:10,(3,4))
```

```
3×4 Matrix{Int64}:
 9   1  9  10
 4  10  8   3
 6   8  3   1
```

```
# Произведение матриц A и B:
A*B
```

```
2×4 Matrix{Int64}:
 150  176  156   97
 148  128  134  102
```

```
# Единичная матрица 3x3:
Matrix{Int}(I, 3, 3)
```

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

```
# Скалярное произведение векторов X и Y:
X = [2, 4, -5]
Y = [1,-1,3]
dot(X,Y)
```

```
-17
```

```
# тоже скалярное произведение:
X'Y
```

```
-17
```

Задачи (метод выполнения)

- Факторизация. Специальные матричные структуры

```
# Задаём квадратную матрицу 3x3 со случайными значениями:
A = rand(3, 3)
```

```
3×3 Matrix{Float64}:
 0.675351  0.298063  0.933472
 0.873562  0.631751  0.686768
 0.264798  0.355774  0.183635
```

```
# Задаём единичный вектор:
x = fill(1.0, 3)
```

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

```
# Задаём вектор b:
b = A*x
```

```
3-element Vector{Float64}:
 1.9068863752704455
 2.192080508690627
 0.8042070810272597
```

```
# Решение исходного уравнения получаем с помощью функции \
# (убеждаемся, что x - единичный вектор):
A\b
```

```
3-element Vector{Float64}:
 1.0000000000000013
 0.9999999999999994
 0.9999999999999993
```

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

```
LU{Float64, Matrix{Float64}, Vector{Int64}}
L factor:
3×3 Matrix{Float64}:
 1.0        0.0       0.0
 0.773101   1.0       0.0
 0.303124  -0.863039  1.0
U factor:
3×3 Matrix{Float64}:
 0.873562   0.631751  0.686768
 0.0       -0.190344  0.402531
 0.0        0.0       0.322859
```

```
# Матрица перестановок:
Alu.P
```

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

```
# Вектор перестановок:
Alu.p
```

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

Задачи (метод выполнения)

- Общая линейная алгебра

```
# Матрица с рациональными элементами:
Arational = Matrix{Rational{BigInt}}(rand(1:10, 3, 3))/10
```

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

```
# Единичный вектор:
x = fill(1, 3)
```

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

```
# Задаём вектор b:
b = Arational*x
```

```
3-element Vector{Rational{BigInt}}:
 17//10
 23//10
  8//5
```

```
# Решение исходного уравнения получаем с помощью функции \
# (убеждаемся, что x - единичный вектор):
Arational\b
```

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

```
# LU-разложение:
lu(Arational)
```

```
LU{Rational{BigInt}, Matrix{Rational{BigInt}}, Vector{Int64}}
L factor:
3×3 Matrix{Rational{BigInt}}:
 1//1    0//1   0//1
 7//10   1//1   0//1
 3//5   -3//14  1//1
U factor:
3×3 Matrix{Rational{BigInt}}:
 1//1  3//5     1//10
 0//1  7//25   83//100
 0//1  0//1   229//280
```

Задачи (метод выполнения)

- Задания для самостоятельного выполнения

```
v = [3, 5, 2, 9]
dot_v = dot(v, v)
```

```
119
```

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

```
4×4 Matrix{Int64}:
  9  15   6  27
 15  25  10  45
  6  10   4  18
 27  45  18  81
```

```
# Left - лево, Right - право
L1 = [1 1; 1 -1]
R1 = [2; 3]
```

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

```
L1\R1
```

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

```
function dia(mat)
    simm = mat + mat'
    razsimm = eigen(simm)
    return inv(razsimm.vectors) * mat * razsimm.vectors
end
```

```
dia (generic function with 1 method)
```

```
mat1 = [1 -2; -2 1]
dia(mat1)
```

```
2×2 Matrix{Float64}:
 -1.0  0.0
  0.0  3.0
```

```
mat2 = [1 -2; -2 3]
dia(mat2)
```

```
2×2 Matrix{Float64}:
 -0.236068     3.46945e-16
  4.44089e-16  4.23607
```

```
mat3 = [1 -2 0; -2 1 2; 0 2 0]
dia(mat3)
```

```
3×3 Matrix{Float64}:
 -2.14134       3.55271e-15  -1.9984e-15
  3.38618e-15   0.515138      1.11022e-16
 -6.66134e-16  -4.44089e-16   3.6262
```

```
([1 -2; -2 1])^10
```

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

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

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

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

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

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

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

```
A = [140 97 74 168 131; 97 106 89 131 36; 74 89 1
val = eigvals(A)
```

```
5-element Vector{Float64}:
 -128.49322764802145
  -55.887784553056875
   42.7521672793189
```

Результаты и их анализ

- Успешно удалось изучить возможностей специализированных пакетов Julia для выполнения и оценки эффективности операций над объектами линейной алгебры.

Благодарю за внимание