РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

дисциплина: Компьютерный практикум по статистическому анализу данных

Студент: Евдокимов Максим Михайлович (1032203019)

Группа: НФИбд-01-20

МОСКВА

2023 г.

Постановка задачи

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

Выполнение работы

1. Произведение векторов

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

```
using LinearAlgebra
# 1.1
v = Vector(1:5); show(v)
dot_v = dot(v, v) # Скалярное произведение
```

```
[1, 2, 3, 4, 5]
55
```

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

```
# 1.2
outer_v = kron(v, v) # Внешнее произведение
reshape(outer_v, (5, 5))
```

```
5×5 Matrix{Int64}:
 1   2   3   4   5
 2   4   6   8  10
 3   6   9  12  15
 4   8  12  16  20
 5  10  15  20  25
```

2. Системы линейных уравнений.

2.1. Решить СЛАУ с двумя неизвестными (a-f):

a) $\begin{cases} x + y = 2, \\ x - y = 3. \end{cases}$ c) $\begin{cases} x + y = 2, \\ 2x + 2y = 5. \end{cases}$ e) $\begin{cases} x + y = 2, \\ 2x + y = 1, \\ x - y = 3. \end{cases}$

b) $\begin{cases} x + y = 2, \\ 2x + 2y = 4. \end{cases}$ d) $\begin{cases} x + y = 1, \\ 2x + 2y = 2, \\ 3x + 3y = 3. \end{cases}$ f) $\begin{cases} x + y = 2, \\ 2x + y = 1, \\ 3x + 2y = 3. \end{cases}$

```julia
function SLOUGH_solving(A, B) # Решение СЛАУ
    try A \ B
    catch e
        return "Нет решения или их бесконечное число."
    else
        return A \ B
    end
end
# 2.1a
m, v = Matrix{Float64}([1 1; 1 -1]), Vector{Float64}([1, 2])
display(SLOUGH_solving(m, v))
#=
using LinearSolve
prob = LinearProblem(m, v)
sol = solve(prob)
sol.u
=#
```

```
2-element Vector{Float64}:
  1.5
 -0.5
```

```julia
# 2.1b
m, v = Matrix{Float64}([1 1; 2 2]), Vector{Float64}([2, 4])
display(SLOUGH_solving(m, v)) # Бесконечно много
```

```
"Нет решения или их бесконечное число."
```

```julia
# 2.1c
m, v = Matrix{Float64}([1 1; 2 2]), Vector{Float64}([2, 5])
display(SLOUGH_solving(m, v)) # Нет решения
```

```
"Нет решения или их бесконечное число."
```

```julia
# 2.1d
m, v = Matrix{Float64}([1 1; 2 2; 3 3]), Vector{Float64}([1, 2, 3])
display(SLOUGH_solving(m, v))
```

```
2-element Vector{Float64}:
 0.49999999999999999
 0.5
```

```julia
# 2.1e
m, v = Matrix{Float64}([1 1; 2 1; 1 -1]), Vector{Float64}([2, 1, 3])
display(SLOUGH_solving(m, v))
```

```
2-element Vector{Float64}:
  1.5000000000000004
 -0.9999999999999997
```

2.2. Решить СЛАУ с тремя неизвестными (a-d):

a) $\begin{cases} x+y+z=2, \\ x-y-2z=3. \end{cases}$

b) $\begin{cases} x+y+z=2, \\ 2x+2y-3z=4, \\ 3x+y+z=1. \end{cases}$

c) $\begin{cases} x+y+z=1, \\ x+y+2z=0, \\ 2x+2y+3z=1. \end{cases}$

d) $\begin{cases} x+y+z=1, \\ x+y+2z=0, \\ 2x+2y+3z=0. \end{cases}$

```
# 2.2a
m, v = Matrix{Float64}([1 1 1; 1 -1 -2; 0 0 1]), Vector{Float64}([2, 3, 2])
display(SLOUGH_solving(m, v))
```

```
3-element Vector{Float64}:
  3.5
 -3.5
  2.0
```

```
# 2.2b
m, v = Matrix{Float64}([1 1 1; 2 2 3; 3 1 1]), Vector{Float64}([2, 4, 1])
display(SLOUGH_solving(m, v))
```

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

```
# 2.2c
m, v = Matrix{Float64}([1 1 1; 1 1 2; 2 2 3]), Vector{Float64}([1, 0, 1])
display(SLOUGH_solving(m, v))
```

```
"Нет решения или их бесконечное число."
```

```
# 2.2d
m, v = Matrix{Float64}([1 1 1; 1 1 2; 2 2 3]), Vector{Float64}([1, 0, 0])
display(SLOUGH_solving(m, v))
```

```
"Нет решения или их бесконечное число."
```

3. Операции с матрицами.

3.1. Приведите приведённые ниже матрицы к диагональному виду:

a) $\begin{pmatrix} 1 & -2 \\ -2 & 1 \end{pmatrix}$ b) $\begin{pmatrix} 1 & -2 \\ -2 & 3 \end{pmatrix}$ c) $\begin{pmatrix} 1 & -2 & 0 \\ -2 & 1 & 2 \\ 0 & 2 & 0 \end{pmatrix}$

```
function todiagonal(M)
    s = size(M)[1]
    ordDown = vcat([[if i == s; [i, j-1] else [i, j] end for j in i+1:s] for i in 1:s]...)
    ordUP = vcat([[if i == s; [j-1, i] else [j, i] end for j in i+1:s] for i in s:-1:1]...)
    res = [[] for _ in 1:s]
    for y in ordDown
        if M[y[2], y[1]] != 0
            coef = M[y[2], y[1]] / M[y[1], y[1]]
            res[y[2]] = [M[y[2], i] - M[y[1], i] * coef for i in 1:s]
        end
    end
    for y in ordUP
        if M[y[2], y[1]] != 0
            coef = M[y[2], y[1]] / M[y[1], y[1]]
            println(coef, " = ", M[y[2], y[1]], " / ", M[y[1], y[1]])
            res[y[2]] = [M[y[2], i] - M[y[1], i] * coef for i in 1:s]
        end
    end
    return res
end

# 3.1a
matr = Matrix{Float64}([1 -2; -2 1])
todiagonal(matr)
```

```
-2.0 = -2.0 / 1.0

2-element Vector{Vector{Any}}:
 [-3.0, 0.0]
 [0.0, -3.0]
```

```
# 3.1b
matr = Matrix{Float64}([1 -2; -2 3])
todiagonal(matr)
```

```
[[1, 2]] [[2, 1]]-2.0
-0.6666666666666666

2-element Vector{Vector{Any}}:
 [-0.33333333333333326, 0.0]
 [0.0, -1.0]
```

```
# 3.1c
matr = Matrix{Float64}([1 -2 0; -2 1 2; 0 2 0])
todiagonal(matr)
```

```
[[1, 2], [1, 3], [2, 3]] [[3, 2], [2, 1], [3, 1]]-2.0
2.0
Inf
-2.0
3-element Vector{Vector{Any}}:
 [-3.0, 0.0, 4.0]
 [NaN, -Inf, NaN]
 [4.0, 0.0, -4.0]
```

3.2. Вычислите:

a) $\begin{pmatrix} 1 & -2 \\ -2 & 1 \end{pmatrix}^{10}$ b) $\sqrt{\begin{pmatrix} 5 & -2 \\ -2 & 5 \end{pmatrix}}$ c) $\sqrt[3]{\begin{pmatrix} 1 & -2 \\ -2 & 1 \end{pmatrix}}$ d) $\sqrt{\begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix}}$

```
# 3.2a
Matrix{Float64}([1 -2; -2 1])^10
```

```
2×2 Matrix{Float64}:
  29525.0  -29524.0
 -29524.0   29525.0
```

```
# 3.2b
A = Matrix{Float64}([5 -2; -2 5])
X = eigvecs(A); display(X)
lamb = Diagonal(eigvals(A)); display(lamb)
lambsqrt = [(l + 0*im)^(1/2) for l in lamb]
Asqrt = X*X^(-1)*lambsqrt
```

```
2×2 Matrix{Float64}:
 -0.707107  -0.707107
 -0.707107   0.707107
2×2 Diagonal{Float64, Vector{Float64}}:
 3.0   ⋅
  ⋅   7.0
2×2 Matrix{ComplexF64}:
 1.73205+0.0im      0.0+0.0im
     0.0+0.0im  2.64575+0.0im
```

```
# 3.2c
A = Matrix{Float64}([1 -2; -2 1])
X = eigvecs(A); display(X)
lamb = Diagonal(eigvals(A)); display(lamb)
lambsqrt = [(l + 0*im)^(1/3) for l in lamb]
Asqrt = X*X^(-1)*lambsqrt
```

```
2×2 Matrix{Float64}:
 -0.707107  -0.707107
 -0.707107   0.707107
2×2 Diagonal{Float64, Vector{Float64}}:
 -1.0   ⋅
   ⋅   3.0

2×2 Matrix{ComplexF64}:
 0.5+0.866025im      0.0+0.0im
 0.0+0.0im       1.44225+0.0im
```

```
# 3.2d
A = Matrix{Float64}([1 2; 2 3])
X = eigvecs(A); display(X)
lamb = Diagonal(eigvals(A)); display(lamb)
lambsqrt = [(l + 0*im)^(1/2) for l in lamb]
Asqrt = X*X^(-1)*lambsqrt
```

```
2×2 Matrix{Float64}:
 -0.850651  0.525731
  0.525731  0.850651
2×2 Diagonal{Float64, Vector{Float64}}:
 -0.236068   ⋅
   ⋅        4.23607

2×2 Matrix{ComplexF64}:
 0.0+0.485868im     1.14251e-16+0.0im
 0.0-2.69711e-17im      2.05817+0.0im
```

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

$$A = \begin{pmatrix} 140 & 97 & 74 & 168 & 131 \\ 97 & 106 & 89 & 131 & 36 \\ 74 & 89 & 152 & 144 & 71 \\ 168 & 131 & 144 & 54 & 142 \\ 131 & 36 & 71 & 142 & 36 \end{pmatrix}$$

```
# 3.3
A = [140 97 74 168 131; 97 106 89 131 36; 74 89 152 144 71; 168 131 144 52 142; 131 36 71 142 36]
println("Первый вариант (полный):"); display(eigen(A))
println("\nВторой вариант (только значения):"); display(eigvals(A))
```

```
Первый вариант (полный):
Eigen{Float64, Float64, Matrix{Float64}, Vector{Float64}}
values:
5-element Vector{Float64}:
 -129.8403784592704
  -56.00818131207872
   42.7506863874373
   87.15844501190563
  541.9394283720052
vectors:
5×5 Matrix{Float64}:
  0.150344   0.646077   0.0107027   0.549067   -0.508322
  0.255635  -0.174498   0.834574   -0.239745   -0.387568
  0.186392   0.238588  -0.422234   -0.731826   -0.441003
 -0.821717  -0.243216  -0.0271033   0.0364277  -0.513387
  0.44954   -0.660346  -0.35264     0.322725   -0.365172

Второй вариант (только значения):
5-element Vector{Float64}:
 -129.84037845927043
  -56.00818131207859
   42.750686387437305
   87.15844501190587
  541.9394283720058
```

3+:

```
# 3.3+
dA = Diagonal(A); buA = Bidiagonal(A, :U); blA = Bidiagonal(A, :L); tA = Tridiagonal(A)
display(dA); display(buA); display(blA); display(tA)
```

```
5×5 Diagonal{Int64, Vector{Int64}}:
 140    ⋅    ⋅   ⋅   ⋅
   ⋅  106    ⋅   ⋅   ⋅
   ⋅    ⋅  152   ⋅   ⋅
   ⋅    ⋅    ⋅  52   ⋅
   ⋅    ⋅    ⋅   ⋅  36
5×5 Bidiagonal{Int64, Vector{Int64}}:
 140   97    ⋅    ⋅    ⋅
   ⋅  106   89    ⋅    ⋅
   ⋅    ⋅  152  144    ⋅
   ⋅    ⋅    ⋅   52  142
   ⋅    ⋅    ⋅    ⋅   36
5×5 Bidiagonal{Int64, Vector{Int64}}:
 140    ⋅    ⋅    ⋅   ⋅
  97  106    ⋅    ⋅   ⋅
   ⋅   89  152    ⋅   ⋅
   ⋅    ⋅  144   52   ⋅
   ⋅    ⋅    ⋅  142  36
5×5 Tridiagonal{Int64, Vector{Int64}}:
 140   97    ⋅    ⋅    ⋅
  97  106   89    ⋅    ⋅
   ⋅   89  152  144    ⋅
   ⋅    ⋅  144   52  142
   ⋅    ⋅    ⋅  142   36
```

4. Линейные модели экономики.

Линейная модель экономики 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.

4.1. 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.

$$\text{a)}\ \begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}\ \text{b)}\ \frac{1}{2}\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}\ \text{c)}\ \frac{1}{10}\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}$$

```
function economicModel(M, y)
    #=
    https://www.hse.ru/data/2014/09/04/1316345039/лекция%204.pdf
    x - Ax = y => (1-A)*x = y => x = (1 - A)^-1 * y
    =#
    x = (Diagonal(fill(1, 2)) - M)^(-1) * y
    return x
end
# 4.1a
A = [1 2; 3 4]; Y = [2; 1]
X = economicModel(A, Y); display(X)
if mapreduce(z -> if z < 0 1 else 0 end, +, X) > 0 println("Непродуктивной.") else println("Продуктивной.") end
```

```
2-element Vector{Float64}:
  0.6666666666666667
 -1.0
Непродуктивной.
```

```
# 4.1b
A = [1 2; 3 4]*0.5; Y = [2; 1]
X = economicModel(A, Y); display(X)
if mapreduce(z -> if z < 0 1 else 0 end, +, X) > 0 println("Непродуктивной.") else println("Продуктивной.") end
```

```
2-element Vector{Float64}:
  0.5
 -1.75
Непродуктивной.
```

```
# 4.1c
A = [1 2; 3 4]*0.1; Y = [2; 5]
X = economicModel(A, Y); display(X)
if mapreduce(z -> if z < 0 1 else 0 end, +, X) > 0 println("Непродуктивной.") else println("Продуктивной.") end
```

```
2-element Vector{Float64}:
  4.583333333333334
 10.625
Продуктивной.
```

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

$$\text{a)}\ \begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}\ \text{b)}\ \frac{1}{2}\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}\ \text{c)}\ \frac{1}{10}\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}$$

```
function OnesModel(M)
    x = (Diagonal(fill(1, size(M, 1))) - M)^(-1)
    return x
end
# 4.2a
A = [1 2; 3 1]
X = OnesModel(A); display(X)
if mapreduce(z -> if z < 0 1 else 0 end, +, X) > 0 println("Непродуктивной.") else println("Продуктивной.") end
```

```
2×2 Matrix{Float64}:
 -0.0  -0.333333
 -0.5   0.0
Непродуктивной.
```

```
# 4.2b
A = [1 2; 3 1]*0.5
X = OnesModel(A); display(X)
if mapreduce(z -> if z < 0 1 else 0 end, +, X) > 0 println("Непродуктивной.") else println("Продуктивной.") end
```

```
2×2 Matrix{Float64}:
 -0.4  -0.8
 -1.2  -0.4
Непродуктивной.
```

```
# 4.2c
A = [1 2; 3 1]*0.1
X = OnesModel(A); display(X)
if mapreduce(z -> if z < 0 1 else 0 end, +, X) > 0 println("Непродуктивной.") else println("Продуктивной.") end
```

```
2×2 Matrix{Float64}:
 1.2  0.266667
 0.4  1.2
Продуктивной.
```

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

a) $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ b) $\frac{1}{2}\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ c) $\frac{1}{10}\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ d) $\begin{pmatrix} 0.1 & 0.2 & 0.3 \\ 0 & 0.1 & 0.2 \\ 0 & 0.1 & 0.3 \end{pmatrix}$

```
eigenvalues(M) = eigen(M).values
# 4.3a
A = [1 2; 3 1]
X = eigenvalues(A); display(X)
if mapreduce(z -> if abs(z) < 1 1 else 0 end, +, X) > 0 return "Продуктивной." else "Непродуктивной." end
```

```
2-element Vector{Float64}:
 -1.4494897427831779
  3.4494897427831783

"Непродуктивной."
```

```
# 4.3b
A = [1 2; 3 1]*0.5
X = eigenvalues(A); display(X)
if mapreduce(z -> if abs(z) < 1 1 else 0 end, +, X) > 0 return "Продуктивной." else "Непродуктивной." end
```

```
2-element Vector{Float64}:
 -0.7247448713915892
  1.724744871391589

"Продуктивной."
```

```
# 4.3c
A = [1 2; 3 1]*0.1
X = eigenvalues(A); display(X)
if mapreduce(z -> if abs(z) < 1 1 else 0 end, +, X) > 0 return "Продуктивной." else "Непродуктивной." end
```

```
2-element Vector{Float64}:
 -0.14494897427831785
  0.34494897427831783

"Продуктивной."
```

```
# 4.3d
A = [0.1 0.2 0.3; 0.0 0.1 0.3; 0.0 0.1 0.3]
X = eigenvalues(A); display(X)
if mapreduce(z -> if abs(z) < 1 1 else 0 end, +, X) > 0 return "Продуктивной." else "Непродуктивной." end
```

```
3-element Vector{Float64}:
 0.0
 0.1
 0.4

"Продуктивной."
```

Выводы

В ходе выполнения лабораторной работы были основные навыки по работе классических (математических) пакетов Julia на примере Linear Algebra. С помощью которого были закреплены навыки по работе с функциями, циклами и различными массивами на примере экономической модели и СЛАУ.