# Лабораторная работа №8

## Оптимизация

Ким Реачна[1]

25 декабря, 2023, Москва, Россия

[1]Российский Университет Дружбы Народов

# Цели и задачи

# Цель лабораторной работы

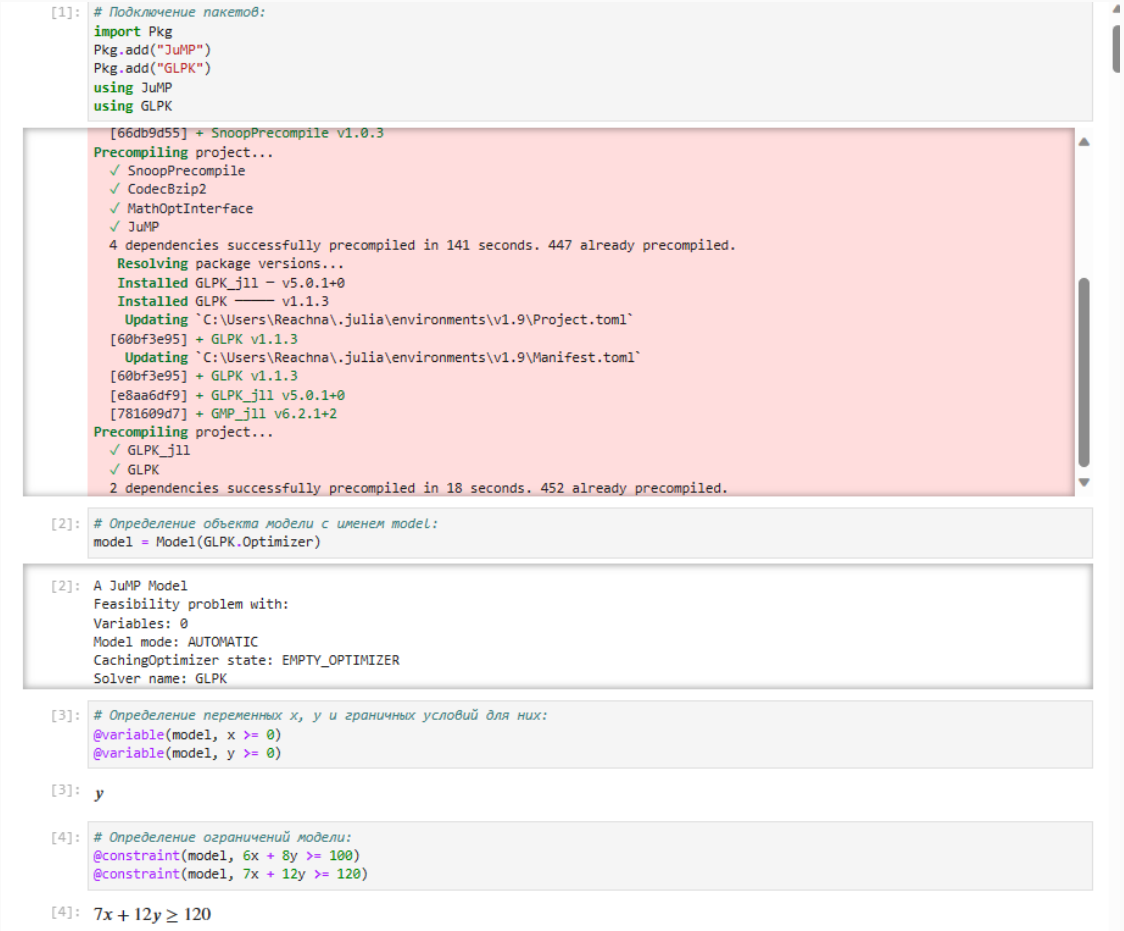
Основная цель работа — освоить пакеты Julia для решения задач оптимизации.

# Задание

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

# Процесс выполнения лабораторной работы

# Линейное программирование



```
[1]: # Подключение пакетов:
import Pkg
Pkg.add("JuMP")
Pkg.add("GLPK")
using JuMP
using GLPK
```

```
  [66db9d55] + SnoopPrecompile v1.0.3
Precompiling project...
  ✓ SnoopPrecompile
  ✓ CodecBzip2
  ✓ MathOptInterface
  ✓ JuMP
  4 dependencies successfully precompiled in 141 seconds. 447 already precompiled.
   Resolving package versions...
   Installed GLPK_jll ─ v5.0.1+0
   Installed GLPK ────── v1.1.3
    Updating `C:\Users\Reachna\.julia\environments\v1.9\Project.toml`
  [60bf3e95] + GLPK v1.1.3
    Updating `C:\Users\Reachna\.julia\environments\v1.9\Manifest.toml`
  [60bf3e95] + GLPK v1.1.3
  [e8aa6df9] + GLPK_jll v5.0.1+0
  [781609d7] + GMP_jll v6.2.1+2
Precompiling project...
  ✓ GLPK_jll
  ✓ GLPK
  2 dependencies successfully precompiled in 18 seconds. 452 already precompiled.
```

```
[2]: # Определение объекта модели с именем model:
model = Model(GLPK.Optimizer)
```

```
[2]: A JuMP Model
Feasibility problem with:
Variables: 0
Model mode: AUTOMATIC
CachingOptimizer state: EMPTY_OPTIMIZER
Solver name: GLPK
```

```
[3]: # Определение переменных x, y и граничных условий для них:
@variable(model, x >= 0)
@variable(model, y >= 0)
```

[3]: $y$

```
[4]: # Определение ограничений модели:
@constraint(model, 6x + 8y >= 100)
@constraint(model, 7x + 12y >= 120)
```

[4]: $7x + 12y \geq 120$

Рис. 1: Примеры линейного программирования

# Векторизованные ограничения и целевая функция оптимизации

```
[9]: # Определение объекта модели с именем vector_model:
vector_model = Model(GLPK.Optimizer)

[9]: A JuMP Model
Feasibility problem with:
Variables: 0
Model mode: AUTOMATIC
CachingOptimizer state: EMPTY_OPTIMIZER
Solver name: GLPK

[10]: # Определение начальных данных:
A= [ 1 1 9 5;
     3 5 0 8;
     2 0 6 13]
b = [7; 3; 5]
c = [1; 3; 5; 2]

[10]: 4-element Vector{Int64}:
 1
 3
 5
 2

[11]: # Определение вектора переменных:
@variable(vector_model, x[1:4] >= 0)

[11]: 4-element Vector{VariableRef}:
 x[1]
 x[2]
 x[3]
 x[4]

[12]: # Определение ограничений модели:
@constraint(vector_model, A * x .== b)

[12]: 3-element Vector{ConstraintRef{Model, MathOptInterface.ConstraintIndex{MathOptInterface.ScalarAffineFunction{Float64}, MathOptInterface.Equa
lTo{Float64}}, ScalarShape}}:
 x[1] + x[2] + 9 x[3] + 5 x[4] == 7
 3 x[1] + 5 x[2] + 8 x[4] == 3
 2 x[1] + 6 x[3] + 13 x[4] == 5

[13]: # Определение целевой функции:
@objective(vector_model, Min, c' * x)

[13]: x[1] + 3 x[2] + 5 x[3] + 2 x[4]

[14]: # Вызов функции оптимизации:
optimize!(vector_model)

[15]: # Определение причины завершения работы оптимизатора:
termination_status(vector_model)

[15]: OPTIMAL::TerminationStatusCode = 1

[16]: # Демонстрация результата оптимизации:
@show objective_value(vector_model);

objective_value(vector_model) = 4.9230769230769225
```

Рис. 2: Векторизованные ограничения и целевая функция оптимизации

# Оптимизация рациона питания

```
[17]: # Контейнер для хранения данных об ограничениях на количество потребляемых калорий, белков, жиров и соли:
category_data = JuMP.Containers.DenseAxisArray(
    [1800 2200;
    91 Inf;
    0 65;
    0 1779],
    ["calories", "protein", "fat", "sodium"],
    ["min", "max"])
```

```
[17]: 2-dimensional DenseAxisArray{Float64,2,...} with index sets:
    Dimension 1, ["calories", "protein", "fat", "sodium"]
    Dimension 2, ["min", "max"]
And data, a 4×2 Matrix{Float64}:
 1800.0  2200.0
   91.0     Inf
    0.0    65.0
    0.0  1779.0
```

```
[18]: # массив данных с наименованиями продуктов:
foods = ["hamburger", "chicken", "hot dog", "fries", "macaroni","pizza","salad", "milk", "ice cream"]
```

```
[18]: 9-element Vector{String}:
 "hamburger"
 "chicken"
 "hot dog"
 "fries"
 "macaroni"
 "pizza"
 "salad"
 "milk"
 "ice cream"
```

```
[19]: # Массив стоимости продуктов:
cost = JuMP.Containers.DenseAxisArray(
[2.49, 2.89, 1.50, 1.89, 2.09, 1.99, 2.49, 0.89, 1.59], foods)
```

```
[19]: 1-dimensional DenseAxisArray{Float64,1,...} with index sets:
    Dimension 1, ["hamburger", "chicken", "hot dog", "fries", "macaroni", "pizza", "salad", "milk", "ice cream"]
And data, a 9-element Vector{Float64}:
 2.49
 2.89
 1.5
 1.89
 2.09
 1.99
 2.49
 0.89
 1.59
```

Рис. 3: Примеры оптимизации рациона питания

# Путешествие по миру

```
[28]: # Подключение пакетов:
import Pkg
Pkg.add("DelimitedFiles")
Pkg.add("CSV")
using DelimitedFiles
using CSV
```

```
  Resolving package versions...
   Updating `C:\Users\Reachna\.julia\environments\v1.9\Project.toml`
 [8bb1440f] + DelimitedFiles v1.9.1
 No Changes to `C:\Users\Reachna\.julia\environments\v1.9\Manifest.toml`
  Resolving package versions...
 No Changes to `C:\Users\Reachna\.julia\environments\v1.9\Project.toml`
 No Changes to `C:\Users\Reachna\.julia\environments\v1.9\Manifest.toml`
```

```
[29]: # Считывание данных:
passportdata = readdlm("passport-index-matrix.csv", ',')
```

```
[29]: 200×200 Matrix{Any}:
 "Passport"                "Albania"          …    "Afghanistan"
 "Afghanistan"             "visa required"         -1
 "Albania"                 -1                      "visa required"
 "Algeria"                 "visa required"         "visa required"
 "Andorra"                 90                      "visa required"
 "Angola"                  "visa required"    …    "visa required"
 "Antigua and Barbuda"     90                      "visa required"
 "Argentina"               90                      "visa required"
 "Armenia"                 90                      "visa required"
 "Australia"               90                      "visa required"
 "Austria"                 90                 …    "visa required"
 "Azerbaijan"              90                      "visa required"
 "Bahamas"                 90                      "visa required"
 ⋮                                            ⋱
 "United Arab Emirates"    90                      "visa required"
 "United Kingdom"          90                      "visa required"
 "United States"           90                 …    "visa required"
 "Uruguay"                 90                      "visa required"
 "Uzbekistan"              "visa required"         "visa required"
 "Vanuatu"                 "visa required"         "visa required"
 "Vatican"                 90                      "visa required"
 "Venezuela"               90                 …    "visa required"
 "Vietnam"                 "visa required"         "visa required"
 "Yemen"                   "visa required"         "visa required"
 "Zambia"                  "visa required"         "visa required"
 "Zimbabwe"                "visa required"         "visa required"
```

```
[30]: # Задаём переменные:
cntr = passportdata[2:end,1]
vf = (x -> typeof(x)==Int64 || x == "VF" || x == "VOA" ? 1 : 0).(passportdata[2:end, 2:end]);
```

Рис. 4: Путешествие по миру

# Портфельные инвестиции

[31]:
```
# Определение объекта модели с именем model:
model = Model(GLPK.Optimizer)
```

[31]:
```
A JuMP Model
Feasibility problem with:
Variables: 0
Model mode: AUTOMATIC
CachingOptimizer state: EMPTY_OPTIMIZER
Solver name: GLPK
```

[32]:
```
# Переменные, ограничения и целевая функция:
@variable(model, pass[1:length(cntr)], Bin)
@constraint(model, [j=1:length(cntr)], sum( vf[i,j]*pass[i] for i in 1:length(cntr)) >= 1)
@objective(model, Min, sum(pass))
```

[32]:
$$pass_1 + pass_2 + pass_3 + pass_4 + pass_5 + pass_6 + pass_7 + pass_8 + pass_9 + pass_{10} + pass_{11} + pass_{12} + pass_{13} + pass_{14} + pass_{15} + pass_{16} + pass_{17} + pass_{18} + pass_{19} + pass_{20} + pass_{21} + pass_{22} + pass_{23} + pass_{24} + pass_{25} + pass_{26} + pass_{27} + pass_{28} + pass_{29} + pass_{30} + [\![\ldots \text{139 terms omitted} \ldots]\!] + pass_{170} + pass_{171} + pass_{172} + pass_{173} + pass_{174} + pass_{175} + pass_{176} + pass_{177} + pass_{178} + pass_{179} + pass_{180} + pass_{181} + pass_{182} + pass_{183} + pass_{184} + pass_{185} + pass_{186} + pass_{187} + pass_{188} + pass_{189} + pass_{190} + pass_{191} + pass_{192} + pass_{193} + pass_{194} + pass_{195} + pass_{196} + pass_{197} + pass_{198} + pass_{199}$$

[33]:
```
# Вызов функции оптимизации:
JuMP.optimize!(model)
termination_status(model)
```

[33]:
```
OPTIMAL::TerminationStatusCode = 1
```

[34]:
```
# Просмотр результата:
print(JuMP.objective_value(model)," passports: ",join(cntr[findall(JuMP.value.(pass) .== 1)],", "))
```

```
63.0 passports: Afghanistan, Andorra, Argentina, Australia, Azerbaijan, Bahrain, Brunei, Cambodia, Cameroon, Canada, Chil
e, Colombia, Comoros, DR Congo, Djibouti, Equatorial Guinea, Eritrea, Fiji, Gabon, Georgia, Guinea, Guinea-Bissau, Hong Ko
ng, Hungary, Indonesia, Iraq, Ireland, Israel, Jamaica, Japan, Kuwait, Laos, Liberia, Libya, Macao, Madagascar, Malaysia,
Maldives, Marshall Islands, Mauritania, Mauritius, Mongolia, Mozambique, Nauru, Nepal, New Zealand, North Korea, Palestin
e, Papua New Guinea, Qatar, Saudi Arabia, Solomon Islands, Somalia, South Sudan, Sri Lanka, Syria, Taiwan, Timor-Leste, To
go, Turkmenistan, United States, Uruguay, Vietnam
```

Рис. 5: Портфельные инвестиции

# Восстановление изображения

[52]:
```
# Подключение необходимых пакетов:
import Pkg
Pkg.add("ImageMagick")
Pkg.add("Convex")
Pkg.add("SCS")
using Images
using Convex
using SCS
```

```
  Resolving package versions...
 No Changes to `C:\Users\Reachna\.julia\environments\v1.9\Project.toml`
 No Changes to `C:\Users\Reachna\.julia\environments\v1.9\Manifest.toml`
  Resolving package versions...
 No Changes to `C:\Users\Reachna\.julia\environments\v1.9\Project.toml`
 No Changes to `C:\Users\Reachna\.julia\environments\v1.9\Manifest.toml`
  Resolving package versions...
 No Changes to `C:\Users\Reachna\.julia\environments\v1.9\Project.toml`
 No Changes to `C:\Users\Reachna\.julia\environments\v1.9\Manifest.toml`
```

[53]:
```
# Считывание исходного изображения:
Kref = load("data/khiam-small.jpg")
```

[53]:

Рис. 6: Примеры восстановления изображений

# Задания для самостоятельного выполнения - Линейное программирование

```
[59]: # Определение объекта модели с именем model:
      model = Model(GLPK.Optimizer)
```

```
[59]: A JuMP Model
      Feasibility problem with:
      Variables: 0
      Model mode: AUTOMATIC
      CachingOptimizer state: EMPTY_OPTIMIZER
      Solver name: GLPK
```

```
[60]: @variable(model, 0 <= x1 <= 10)
      @variable(model, x2 >= 0)
      @variable(model, x3 >= 0)
```

[60]: $x3$

```
[61]: @constraint(model, -x1 + x2 + 3x3 <= -5)
      @constraint(model, x1 + 3x2 - 7x3 <= 10)
```

[61]: $x1 + 3x2 - 7x3 \leq 10$

```
[62]: @objective(model, Max, x1 + 2x2 + 5x3)
```

[62]: $x1 + 2x2 + 5x3$

```
[63]: optimize!(model)
```

```
[64]: termination_status(model)
```

```
[64]: OPTIMAL::TerminationStatusCode = 1
```

```
[65]: @show value(x1);
      @show value(x2);
      @show value(x3);

      @show objective_value(model);
```

```
value(x1) = 10.0
value(x2) = 2.1875
value(x3) = 0.9375
objective_value(model) = 19.0625
```

Рис. 7: Линейное программирование

# Линейное программирование. Использование массивов

```
[66]: vector_model_2 = Model(GLPK.Optimizer)

[66]: A JuMP Model
      Feasibility problem with:
      Variables: 0
      Model mode: AUTOMATIC
      CachingOptimizer state: EMPTY_OPTIMIZER
      Solver name: GLPK

[67]: A = [-1 1 3;
           1 3 -7]
      b = [-5; 10]
      c = [1; 2; 5]

[67]: 3-element Vector{Int64}:
       1
       2
       5

[68]: @variable(vector_model_2, x[1:3] >= 0)
      set_upper_bound(x[1], 10)

[69]: @constraint(vector_model_2, A * x .== b)

[69]: 2-element Vector{ConstraintRef{Model, MathOptInterface.ConstraintIndex{MathOptInterface.ScalarAffineFunction{Float64}, Mat
      hOptInterface.EqualTo{Float64}}, ScalarShape}}:
       -x[1] + x[2] + 3 x[3] == -5
       x[1] + 3 x[2] - 7 x[3] == 10

[70]: @objective(vector_model_2, Max, c' * x)

[70]: x_1 + 2x_2 + 5x_3

[71]: optimize!(vector_model_2)

[72]: termination_status(vector_model_2)

[72]: OPTIMAL::TerminationStatusCode = 1

[73]: @show value(x1);
      @show value(x2);
      @show value(x3);

      @show objective_value(vector_model_2);

      value(x1) = 10.0
      value(x2) = 2.1875
      value(x3) = 0.9375
      objective_value(vector_model_2) = 19.0625
```

Рис. 8: Линейное программирование. Использование массивов

# Выпуклое программирование

```
[74]: using Convex
      using SCS
```

```
[75]: m = 5
      n = 4

      A = rand(m, n)
      b = rand(m)

      display(A)
      println()
      display(b)
```

```
5×4 Matrix{Float64}:
 0.849155  0.936487  0.74695   0.513831
 0.401326  0.875853  0.874495  0.631306
 0.800074  0.526601  0.666555  0.145633
 0.483516  0.305902  0.636353  0.582247
 0.336612  0.354855  0.968644  0.92455

5-element Vector{Float64}:
 0.45183142669440834
 0.6667329596934422
 0.7275304988925214
 0.8926481804606815
 0.3173554782366863
```

```
[76]: x = Variable(n)
      display(x)
```

```
Variable
size: (4, 1)
sign: real
vexity: affine
id: 173…816
```

```
[77]: model = minimize(Convex.sumsquares(A*x - b), [x >= 0])
```

```
[77]: minimize
      └─ qol_elem (convex; positive)
         ├─ norm2 (convex; positive)
         │  └─ + (affine; real)
         │     ├─ …
         │     └─ …
         └─ [1.0;;]
      subject to
      └─ >= constraint (affine)
         ├─ 4-element real variable (id: 173…816)
         └─ 0

      status: `solve!` not called yet
```

Рис. 9: Выпуклое программирование

# Оптимальная рассадка по залам

```julia
# Условия для учета приоритетов
for j in 1:num_sections
    for k in 1:3
        @constraint(model, sum(x[i, j] for i in findall(priorities[:, j] .== k)) == 0)
    end
end

# Условие для третьей секции, где нужно ровно 220 человек
@constraint(model, sum(x[i, 3] for i in 1:num_participants) == target_capacity)

# Функция цели: максимизация общего числа посетителей
@objective(model, Max, sum(x))

# Решение задачи
optimize!(model)

# Вывод результатов
println("Status: ", termination_status(model))

if termination_status(model) == MOI.OPTIMAL
    println("Objective value: ", objective_value(model))

    allocation = argmax(value.(x), dims=2)
    for i in 1:num_participants
        println("Слушатель $i посещает секцию $(allocation[i])")
    end
else
    println("Решение не найдено")
end
```

```
GLPK Simplex Optimizer 5.0
1021 rows, 5000 columns, 16000 non-zeros
      0: obj = -0.000000000e+000 inf =  1.220e+003 (1001)
     15: obj = -0.000000000e+000 inf =  1.220e+003 (1001)
LP HAS NO PRIMAL FEASIBLE SOLUTION
GLPK Integer Optimizer 5.0
1021 rows, 5000 columns, 16000 non-zeros
5000 integer variables, all of which are binary
Preprocessing...
PROBLEM HAS NO PRIMAL FEASIBLE SOLUTION
Status: INFEASIBLE
Решение не найдено
```

Рис. 10: Оптимальная рассадка по залам

# План приготовления кофе

```
[82]: using JuMP
      using GLPK
```

```
[83]: coffee_type = ["Raf coffee", "Capuccino"]

      balance_data = JuMP.Containers.DenseAxisArray(
          [40 140 5;
          30 120 0],
          coffee_type,
          ["beans", "milk", "sugar"])
```

```
[83]: 2-dimensional DenseAxisArray{Int64,2,...} with index sets:
          Dimension 1, ["Raf coffee", "Capuccino"]
          Dimension 2, ["beans", "milk", "sugar"]
      And data, a 2×3 Matrix{Int64}:
       40  140  5
       30  120  0
```

```
[84]: coffee_data = JuMP.Containers.DenseAxisArray(
          [0 500;
          0 2000;
          40 40],
          ["beans", "milk", "sugar"],
          ["min", "max"])
```

```
[84]: 2-dimensional DenseAxisArray{Int64,2,...} with index sets:
          Dimension 1, ["beans", "milk", "sugar"]
          Dimension 2, ["min", "max"]
      And data, a 3×2 Matrix{Int64}:
        0   500
        0  2000
       40    40
```

```
[85]: price_coffee = JuMP.Containers.DenseAxisArray([400, 300], coffee_type)
```

```
[85]: 1-dimensional DenseAxisArray{Int64,1,...} with index sets:
          Dimension 1, ["Raf coffee", "Capuccino"]
      And data, a 2-element Vector{Int64}:
       400
       300
```

```
[86]: ingredients = ["beans", "milk", "sugar"]
```

```
[86]: 3-element Vector{String}:
       "beans"
       "milk"
       "sugar"
```

```
[87]: model = Model(GLPK.Optimizer)
```

```
[87]: A JuMP Model
      Feasibility problem with:
      Variables: 0
      Model mode: AUTOMATIC
      CachingOptimizer state: EMPTY_OPTIMIZER
```

Рис. 11: План приготовления кофе

# Выводы по проделанной работе

# Вывод

Освоила пакеты Julia для решения задач оптимизации.