МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ»
Факультет физико-математических и естественных наук

ОТЧЕТ

По лабораторной работе №8

Оптимизация

Выполнил:
Студент группы: НПИбд-02-21
Студенческий билет: № 1032217060
ФИО студента: Королев Адам Маратович

Москва 2025

Цели работы:

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

Выполнение работы:

1. Используя Jupyter Lab, повторите примеры из раздела 8.2.

```
# ------------------------ Раздел 8.2 Примеры ------------------------

# Пример 1: Линейное программирование
println("Пример 1: Линейное программирование")
model1 = Model(GLPK.Optimizer)
@variable(model1, x >= 0)
@variable(model1, y >= 0)
@constraint(model1, 6x + 8y >= 100)
@constraint(model1, 7x + 12y >= 120)
@objective(model1, Min, 12x + 20y)
optimize!(model1)
println("Результаты Пример 1:")
println("x = ", value(x))
println("y = ", value(y))
println("Objective value = ", objective_value(model1))

# Пример 2: Векторизованные ограничения
println("\nПример 2: Векторизованные ограничения")
A = [1 1 9 5; 3 5 0 8; 2 0 6 13]
b = [7; 3; 5]
c = [1; 3; 5; 2]
model2 = Model(GLPK.Optimizer)
@variable(model2, x[1:4] >= 0)
@constraint(model2, A * x .== b)
@objective(model2, Min, c' * x)
optimize!(model2)
println("Результаты Пример 2:")
println("x = ", value.(x))
println("Objective value = ", objective_value(model2))

# Пример 3: Оптимизация рациона питания (исправленный)
println("\nПример 3: Оптимизация рациона питания")

# Определение данных
```

```
# Пример 3: Оптимизация рациона питания (исправленный)
println("\nПример 3: Оптимизация рациона питания")

# Определение данных
foods = ["hamburger", "chicken", "hot dog", "fries", "macaroni", "pizza", "salad", "milk", "ice cream"]
cost = [2.49, 2.89, 1.50, 1.89, 2.09, 1.99, 2.49, 0.89, 1.59]
food_data = [
    410 24 26 730;
    420 32 10 1190;
    560 20 32 1800;
    380 4 19 270;
    320 12 10 930;
    320 15 12 820;
    320 31 12 1230;
    100 8 2.5 125;
    330 8 10 180
]
categories = ["calories", "protein", "fat", "sodium"]
category_data = [
    1800 2200;  # Минимум и максимум калорий
    91 Inf;     # Минимум и максимум белков
    0 65;       # Минимум и максимум жиров
    0 1779      # Минимум и максимум соли
]

# Индексируем category_data по номерам
category_min = category_data[:, 1]
category_max = category_data[:, 2]

# Создание модели
model3 = Model(GLPK.Optimizer)

# Переменные для количества покупаемых продуктов
```

```
Пример 1: Линейное программирование
Результаты Пример 1:
x = 14.999999999999993
y = 1.2500000000000047
Objective value = 205.0

Пример 2: Векторизованные ограничения
Результаты Пример 2:
x = [0.4230769230769232, 0.34615384615384615, 0.6923076923076922, 0.0]
Objective value = 4.9230769230769225

Пример 3: Оптимизация рациона питания
Результаты Пример 3:
Купить:
  hamburger: 0.6045138888888921
  chicken: 0.0
  hot dog: 0.0
  fries: 0.0
  macaroni: 0.0
  pizza: 0.0
  salad: 0.0
  milk: 6.970138888888879
  ice cream: 2.5913194444444416
Общая стоимость: 11.828861111111106
```

2. Выполните задания для самостоятельной работы (раздел 8.4).

```julia
# ------------------------ Раздел 8.4 Задания ------------------------

# Задание 8.4.1: Линейное программирование
println("\nЗадание 8.4.1: Линейное программирование")
model4 = Model(GLPK.Optimizer)
@variable(model4, 0 <= x1 <= 10)
@variable(model4, x2 >= 0)
@variable(model4, x3 >= 0)
@constraint(model4, -x1 + x2 + 3x3 <= -5)
@constraint(model4, x1 + 3x2 - 7x3 <= 10)
@objective(model4, Max, x1 + 2x2 + 5x3)
optimize!(model4)
println("Результаты Задание 8.4.1:")
println("x1 = ", value(x1))
println("x2 = ", value(x2))
println("x3 = ", value(x3))
println("Objective value = ", objective_value(model4))

# Задание 8.4.2: Линейное программирование с массивами
println("\nЗадание 8.4.2: Линейное программирование с массивами")
A_task2 = [-1 1 3; 1 3 -7]
b_task2 = [-5; 10]
c_task2 = [1, 2, 5]
model5 = Model(GLPK.Optimizer)
@variable(model5, x[1:3] >= 0)
@constraint(model5, x[1] <= 10)
@constraint(model5, A_task2 * x .<= b_task2)
@objective(model5, Max, c_task2' * x)
optimize!(model5)
println("Результаты Задание 8.4.2:")
println("x = ", value.(x))
println("Objective value = ", objective_value(model5))
```

```julia
# Задание 8.4.3: Выпуклое программирование
println("\n3адание 8.4.3: Выпуклое программирование")
m, n = 4, 3
A_rand = rand(m, n)
b_rand = rand(m)
x_var = Variable(n)
problem = minimize(sumsquares(A_rand * x_var - b_rand), x_var >= 0)
solve!(problem, SCS.Optimizer)
println("Результаты Задание 8.4.3:")
println("x = ", evaluate(x_var))

# Задание 8.4.4: Оптимальная рассадка по залам
println("\n3адание 8.4.4: Оптимальная рассадка по залам")
num_sections = 5
num_participants = 1000
priorities = rand(1:3, num_participants, num_sections)
priorities[:, 3] .= 10000 # Условия для третьей секции
min_people = [180, 180, 220, 180, 180]
max_people = [250, 250, 220, 250, 250]
model6 = Model(GLPK.Optimizer)
@variable(model6, assign[1:num_participants, 1:num_sections] >= 0, Bin)
@constraint(model6, [s in 1:num_sections],
    min_people[s] <= sum(assign[:, s]) <= max_people[s])
@constraint(model6, [p in 1:num_participants], sum(assign[p, :]) == 1)
@objective(model6, Min, sum(priorities .* assign))
optimize!(model6)
println("Результаты Задание 8.4.4:")
println("Распределение по секциям:")
for s in 1:num_sections
    println("Секция $s: ", sum(value.(assign[:, s])))
end

# Задание 8.4.5: План приготовления кофе
println("\n3адание 8.4.5: План приготовления кофе")
```

```
Задание 8.4.1: Линейное программирование
Результаты Задание 8.4.1:
x1 = 10.0
x2 = 2.1875
x3 = 0.9375
Objective value = 19.0625

Задание 8.4.2: Линейное программирование с массивами
Результаты Задание 8.4.2:
x = [10.0, 2.1875, 0.9375]
Objective value = 19.0625

Задание 8.4.3: Выпуклое программирование
------------------------------------------------------------------
               SCS v3.2.7 - Splitting Conic Solver
        (c) Brendan O'Donoghue, Stanford University, 2012
------------------------------------------------------------------
problem:  variables n: 4, constraints m: 9
cones:    l: linear vars: 3
          q: soc vars: 6, qsize: 1
settings: eps_abs: 1.0e-004, eps_rel: 1.0e-004, eps_infeas: 1.0e-007
          alpha: 1.50, scale: 1.00e-001, adaptive_scale: 1
          max_iters: 100000, normalize: 1, rho_x: 1.00e-006
          acceleration_lookback: 10, acceleration_interval: 10
          compiled with openmp parallelization enabled
lin-sys:  sparse-direct-amd-qdldl
          nnz(A): 17, nnz(P): 0
------------------------------------------------------------------
 iter | pri res | dua res |   gap   |   obj   |  scale  | time (s)
------------------------------------------------------------------
     0|1.20e+001 1.00e+000 1.66e+001 -8.10e+000 1.00e-001 2.78e-003
   125|6.47e-007 2.00e-007 1.28e-006 5.35e-001 7.54e-001 3.20e-003
------------------------------------------------------------------
status:  solved
timings: total: 3.20e-003s = setup: 7.13e-005s + solve: 3.13e-003s
         lin-sys: 2.40e-005s  cones: 1.55e-005s  accel: 4.00e-006s
```

```
          compiled with openmp parallelization enabled
lin-sys:  sparse-direct-amd-qdldl
          nnz(A): 17, nnz(P): 0
-----------------------------------------------------------------
 iter | pri res | dua res |   gap   |   obj   |  scale  | time (s)
-----------------------------------------------------------------
     0|1.20e+001 1.00e+000 1.66e+001 -8.10e+000 1.00e-001 2.78e-003
   125|6.47e-007 2.00e-007 1.28e-006 5.35e-001 7.54e-001 3.20e-003
-----------------------------------------------------------------
status:  solved
timings: total: 3.20e-003s = setup: 7.13e-005s + solve: 3.13e-003s
         lin-sys: 2.40e-005s, cones: 1.55e-005s, accel: 4.00e-006s
-----------------------------------------------------------------
objective = 0.535499
-----------------------------------------------------------------
Результаты Задание 8.4.3:
[ Info: [Convex.jl] Compilation finished: 9.64 seconds, 1.062 GiB of memory allocated

x = [1.0065286731881893, -4.089043566832975e-8, 0.13587243914115946]

Задание 8.4.4: Оптимальная рассадка по залам
Результаты Задание 8.4.4:
Распределение по секциям:
Секция 1: 240.0
Секция 2: 180.0
Секция 3: 220.0
Секция 4: 180.0
Секция 5: 180.0

Задание 8.4.5: План приготовления кофе
Результаты Задание 8.4.5:
Raf кофе = 8.0
Капучино = 6.0
Общая выручка = 5000.0
```

Выводы:

В процессе выполнения работы мною были освоены пакеты Julia для решения задач оптимизации.