ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Профессор |  |  |  | С.И. Колесникова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1  Нелинейное программирование. Вариационный принцип. |
| **по дисциплине: Компьютерное моделирование** |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 4134к |  |  |  | Костяков Н.А. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

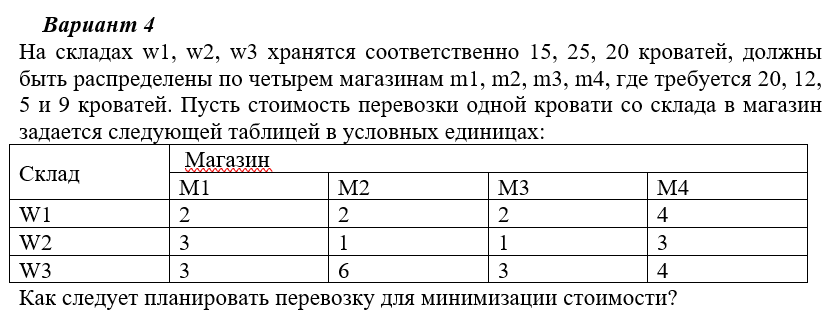
Санкт-Петербург

2024

# Цель работы.

Цель настоящей работы – освоить средства моделирования задач линейного программирования. Решение простейшей вариационной задачи

# Постановка задачи



# Формализованная постановка задачи

|  |
| --- |
|  |

Обозначим переменные ( x{ij} ) как количество кроватей, которые нужно перевезти с склада ( Wi ) в магазин ( Mj ).

1. (x{11} ): количество кроватей из склада ( W1 ) в магазин ( M1 ).
2. (x{12} ): количество кроватей из склада ( W1 ) в магазин ( M2 ).
3. И так далее для всех комбинаций ( i ) и ( j ) (где ( i \in {1, 2, 3} ) и ( j \in {1, 2, 3, 4} )).

### **Целевая функция**

Цель — минимизировать общую стоимость транспортировки, которая выражается как сумма произведений количества кроватей и стоимости транспортировки для каждой пары склад-магазин:

[ \text{Minimize } Z = 2x\_{11} + 2x\_{12} + 2x\_{13} + 4x\_{14} + 3x\_{21} + 1x\_{22} + 1x\_{23} + 3x\_{24} + 3x\_{31} + 6x\_{32} + 3x\_{33} + 4x\_{34} ]

### **Ограничения**

1. **Ограничения на потребности магазинов**:
2. ( x\_{11} + x\_{21} + x\_{31} = 20 ) (для ( M1 ))
3. ( x\_{12} + x\_{22} + x\_{32} = 12 ) (для ( M2 ))
4. ( x\_{13} + x\_{23} + x\_{33} = 5 ) (для ( M3 ))
5. ( x\_{14} + x\_{24} + x\_{34} = 9 ) (для ( M4 ))
6. **Ограничения на запасы складов**:
7. (x{11} + x\_{12} + x\_{13} + x\_{14} <= 15 ) (для ( W1 ))
8. ( x\_{21} + x\_{22} + x\_{23} + x\_{24} <= 25 ) (для ( W2 ))
9. ( x\_{31} + x\_{32} + x\_{33} + x\_{34} <= 20 ) (для ( W3 ))
10. **Ограничения на неотрицательность**:
11. ( x\_{ij} >= 0 ) для всех ( i ) и ( j ).

### **Итоговая формулировка задачи**

Мы должны минимизировать функцию стоимости ( Z ), удовлетворяя ограничениям по потребностям магазинов и доступным запасам на складах.

# Скриншоты решения Excell

Я использовал LibreOffice Calc так как писал работу из под дистрибутива на linux

# 

Рисунок 1 – Результаты решения

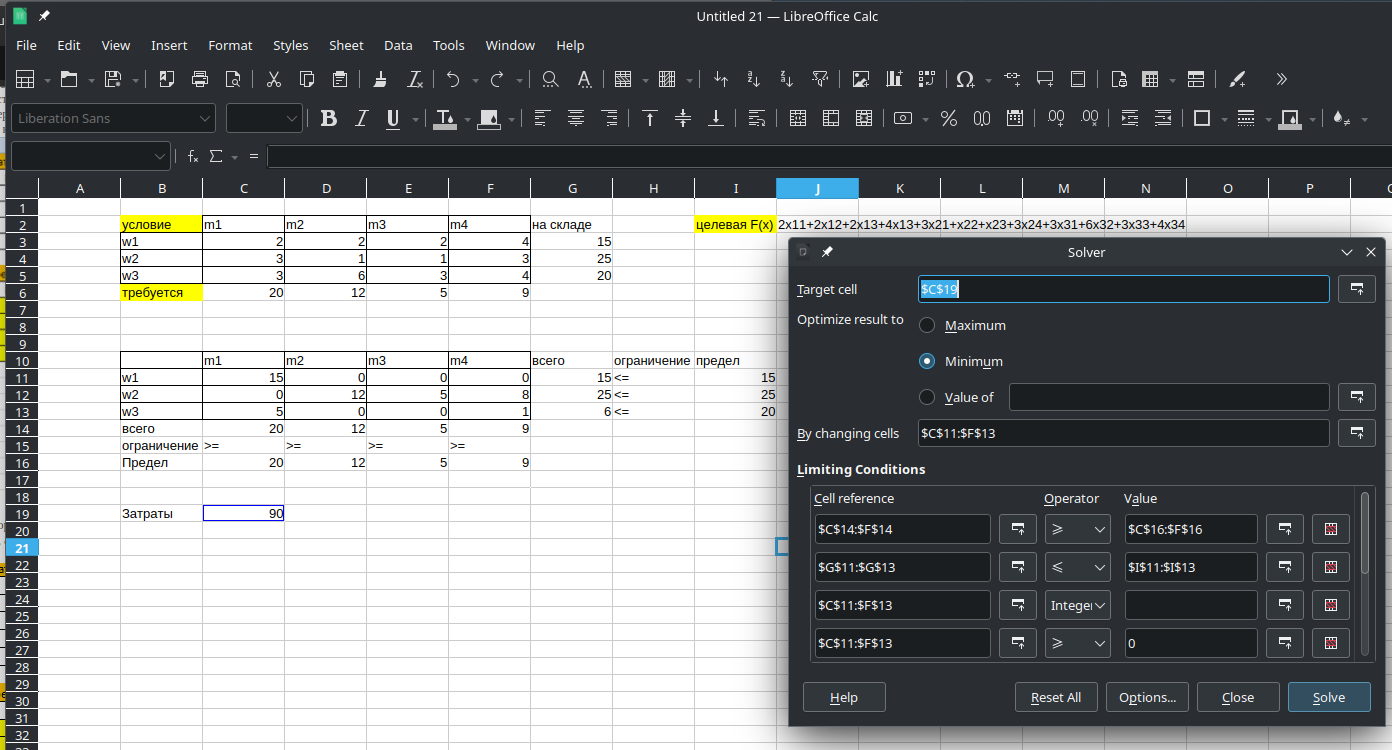
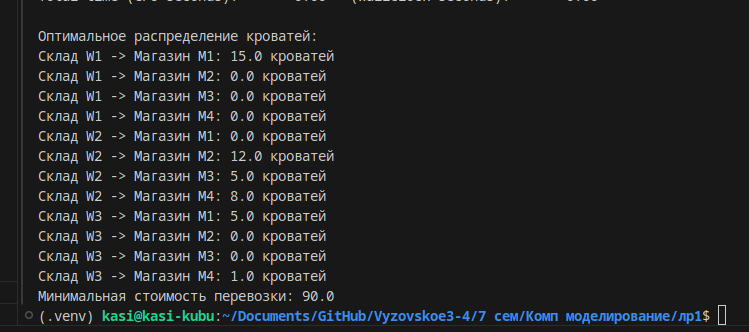


Рисунок 2 – настройка для решения задачи

# Скриншоты работы программы на python



# Листинг программы

import pulp

supply = [15, 25, 20]  # w1, w2, w3

warehouses = ['W1', 'W2', 'W3']

demand = [20, 12, 5, 9]  # m1, m2, m3, m4

stores = ['M1', 'M2', 'M3', 'M4']

cost = {

    ('W1', 'M1'): 2, ('W1', 'M2'): 2, ('W1', 'M3'): 2, ('W1', 'M4'): 4,

    ('W2', 'M1'): 3, ('W2', 'M2'): 1, ('W2', 'M3'): 1, ('W2', 'M4'): 3,

    ('W3', 'M1'): 3, ('W3', 'M2'): 6, ('W3', 'M3'): 3, ('W3', 'M4'): 4,

}

problem = pulp.LpProblem("Transportation\_Problem", pulp.LpMinimize)

x = pulp.LpVariable.dicts("x", [(w, s) for w in warehouses for s in stores], lowBound=0, cat='Continuous')

problem += pulp.lpSum(cost[(w, s)] \* x[(w, s)] for w in warehouses for s in stores), "Total\_Cost"

for i, w in enumerate(warehouses):

    problem += pulp.lpSum(x[(w, s)] for s in stores) <= supply[i], f"Supply\_Constraint\_{w}"

for j, s in enumerate(stores):

    problem += pulp.lpSum(x[(w, s)] for w in warehouses) >= demand[j], f"Demand\_Constraint\_{s}"

problem.solve()

print("Оптимальное распределение кроватей:")

for w in warehouses:

    for s in stores:

        print(f"Склад {w} -> Магазин {s}: {x[(w, s)].varValue} кроватей")

    print()

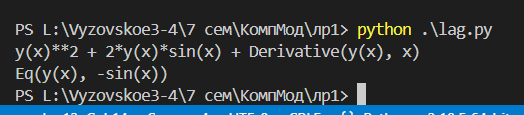
print("Минимальная стоимость перевозки:", pulp.value(problem.objective))

# Часть 2 – решение вариационной задачи

**Вариант 4**



# Скриншот решения на python



# Листинг программы

# this code requires SymPy v1.4

# следующие две строки не нужны при запуске кода на live.sympy.org

from sympy import init\_printing, sin

import numpy as np

init\_printing()

from sympy import Symbol, Function, Derivative, dsolve, solve

x = Symbol('x')

y = Function('y')(x)

dy = Derivative(y)

F = (y-(1/2)\*y\*\*2)\*sin(x)

F.doit() # выводиим выражение в человекочитаемом формате ...

print(F) # ... и в машиночитаемом виде

dFdy = Derivative(F, y)

dFd1y = Derivative(F, dy)

dFdy.doit()

dFd1y.doit()

L = dFdy - Derivative(dFd1y, x)

sol = dsolve(L)

eq1 = sol.subs({x:np.pi/4, y:-np.log(np.sqrt(2))})

eq2 = sol.subs({x:np.pi/2, y:0})

coeffs = solve([eq1, eq2])

res = sol.subs(coeffs)

print(res.doit())