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 |  |  |  | Столяров Н.С. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

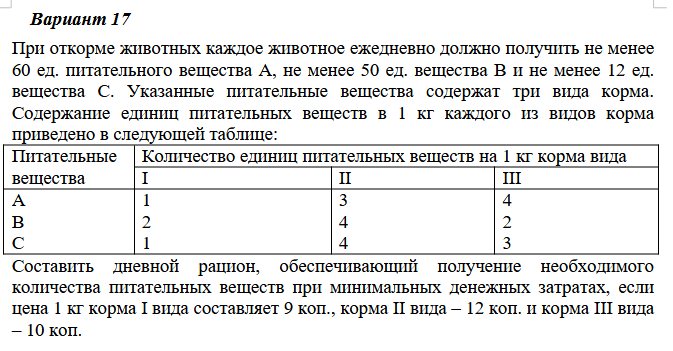
Санкт-Петербург

2024

# Цель работы.

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

# Постановка задачи



# Формализованная постановка задачи

|  |
| --- |
|  |

Обозначим переменные (xi) как количество килограммов корма iii-го вида, которое необходимо использовать в дневном рационе.

1. x1: количество килограммов корма I вида.
2. x2: количество килограммов корма II вида.
3. x3: количество килограммов корма III вида.

### **Целевая функция**

Цель — минимизировать общую стоимость кормов, которая выражается как:

Minimize Z=9x1+12x2+10x3

где

* 9 — цена 1 кг корма I вида,
* 12 — цена 1 кг корма II вида,
* 10 — цена 1 кг корма III вида.

### **Ограничения**

Рацион должен удовлетворять минимальным требованиям к количеству питательных веществ A,B,CA, B, CA,B,C:

1. Ограничение по веществу A: x1+3x2+4x3≥60
2. Ограничение по веществу B: 2x1+4x2+2x3≥50
3. Ограничение по веществу C: x1+4x2+3x3≥12
4. Ограничения на неотрицательность: x1≥0,x2≥0,x3≥0

### **Итоговая формулировка задачи**

Мы должны **минимизировать** функцию стоимости: Z=9x1+12x2+10x3   
при выполнении ограничений:

* x1+3x2+4x3≥60.
* 2x1+4x2+2x3≥50.
* 1x1+4x2+3x3≥12.
* x1≥0,x2≥0,x3≥0.

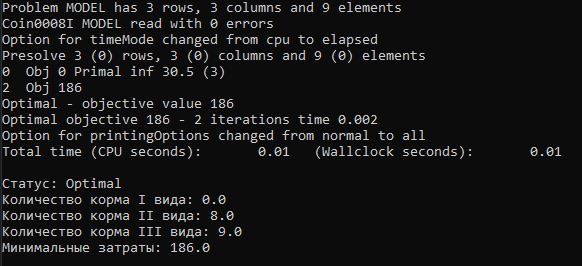
# Скриншоты решения Excell

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

# 

Рисунок 1 – Результаты решения

# Скриншоты работы программы на python



# Листинг программы

|  |
| --- |
| from pulp import LpProblem, LpMinimize, LpVariable, lpSum, LpStatus, value  # Содержание питательных веществ в 1 кг каждого вида корма  feed\_nutrients = {  "I": [1, 2, 1], # [A, B, C]  "II": [3, 4, 4],  "III": [4, 2, 3]  }  # Цены на 1 кг каждого вида корма  feed\_prices = {  "I": 9,  "II": 12,  "III": 10  }  # Минимальные требования к питательным веществам  min\_requirements = {  "A": 60,  "B": 50,  "C": 12  }  # Создание задачи  problem = LpProblem("Animal\_Feed\_Optimization", LpMinimize)  # Переменные  x1 = LpVariable("Feed\_Type\_I", lowBound=0)  x2 = LpVariable("Feed\_Type\_II", lowBound=0)  x3 = LpVariable("Feed\_Type\_III", lowBound=0)  # Целевая функция  problem += feed\_prices["I"] \* x1 + feed\_prices["II"] \* x2 + feed\_prices["III"] \* x3  # Ограничения  problem += feed\_nutrients["I"][0] \* x1 + feed\_nutrients["II"][0] \* x2 + feed\_nutrients["III"][0] \* x3 >= min\_requirements["A"] # A  problem += feed\_nutrients["I"][1] \* x1 + feed\_nutrients["II"][1] \* x2 + feed\_nutrients["III"][1] \* x3 >= min\_requirements["B"] # B  problem += feed\_nutrients["I"][2] \* x1 + feed\_nutrients["II"][2] \* x2 + feed\_nutrients["III"][2] \* x3 >= min\_requirements["C"] # C  # Решение задачи  problem.solve()  # Результаты  print("Статус:", LpStatus[problem.status])  print("Количество корма I вида:", value(x1))  print("Количество корма II вида:", value(x2))  print("Количество корма III вида:", value(x3))  print("Минимальные затраты:", value(problem.objective)) |

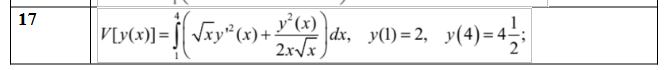
# Скриншоты работы программы на matlab

# Листинг программы

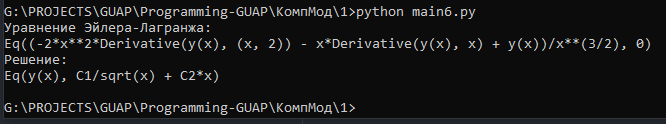
|  |
| --- |
| % Содержание питательных веществ в 1 кг каждого вида корма  feed\_nutrients = [  1, 3, 4; % A  2, 4, 2; % B  1, 4, 3 % C  ];  % Цены на 1 кг каждого вида корма  feed\_prices = [9; 12; 10];  % Минимальные требования к питательным веществам  min\_requirements = [60; 50; 12];  % Определение коэффициентов целевой функции  f = feed\_prices;  % Определение матрицы ограничений  A = -feed\_nutrients; % Умножаем на -1 для преобразования в стандартный вид  b = -min\_requirements;  % Ограничения на неотрицательность  lb = [0; 0; 0]; % Нижние границы для переменных  % Решение задачи линейного программирования  options = optimoptions('linprog', 'Display', 'off');  [x, fval, exitflag] = linprog(f, A, b, [], [], lb, [], options);  % Проверка статуса решения  if exitflag == 1  fprintf('Статус: Успешно\n');  fprintf('Количество корма I вида: %.2f\n', x(1));  fprintf('Количество корма II вида: %.2f\n', x(2));  fprintf('Количество корма III вида: %.2f\n', x(3));  fprintf('Минимальные затраты: %.2f\n', fval);  else  fprintf('Статус: Не удалось найти решение\n');  end |

# Часть 2 – решение вариационной задачи

**Вариант 17**



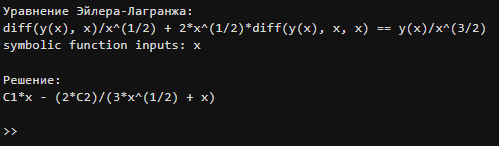
# Скриншот решения на python



# Листинг программы

|  |
| --- |
| from sympy import symbols, Function, sqrt, diff, Eq, simplify, dsolve, solve  # Определение переменных  x = symbols('x')  y = Function('y')(x)  # Первая производная y по x  dy = diff(y, x)  # Определение лагранжиана  F = sqrt(x) \* dy\*\*2 + (y\*\*2) / (2 \* x \* sqrt(x))  # Частные производные  dFdy = diff(F, y) # Производная по y  dFd1y = diff(F, dy) # Производная по y'  # Производная по x от dF/dy'  d\_dFd1y\_dx = diff(dFd1y, x)  # Уравнение Эйлера-Лагранжа  L = dFdy - d\_dFd1y\_dx # Левое выражение уравнения Эйлера-Лагранжа  # Приведение уравнения к стандартному виду  eqn = Eq(simplify(L), 0)  print("Уравнение Эйлера-Лагранжа:")  print(eqn)  # Решение уравнения  sol = dsolve(eqn)  print("Решение:")  print(sol)  eq1 = sol.subs({x:2, y:1})  eq2 = sol.subs({x:4 \* (1/2), y:4})  coeffs = solve([eq1, eq2])  res = sol.subs(coeffs)  res.doit() |

# Скриншот решения на matlab



# Листинг программы

|  |
| --- |
| % Очистка рабочей области и консоли  clear all  clc  % Объявляем переменные  syms x y(x)  % Первая производная y по x  dy = diff(y, x);  % Определение лагранжиана  F = sqrt(x) \* dy^2 + (y^2) / (2 \* x \* sqrt(x));  % Частные производные  dFdy = diff(F, y); % Производная по y  dFd1y = diff(F, dy); % Производная по y'  % Производная по x от dF/dy'  d\_dFd1y\_dx = diff(dFd1y, x);  % Уравнение Эйлера-Лагранжа  L = dFdy - d\_dFd1y\_dx; % Левое выражение уравнения Эйлера-Лагранжа  % Приведение уравнения к стандартному виду  eqn = simplify(L == 0);  disp('Уравнение Эйлера-Лагранжа:');  disp(eqn);  % Решение уравнения  % sol = dsolve(eqn, y);  % disp('Решение:');  % disp(sol);  % Определяем переменные  syms y(x)  % Уравнение  eqn = diff(y, x) / sqrt(x) + 2 \* sqrt(x) \* diff(y, x, x) == y / x^(3/2);  % disp('Уравнение Эйлера-Лагранжа:');  % disp(eqn);  % Решение уравнения  sol = dsolve(eqn);  disp('Решение:');  disp(sol); |