ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

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

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

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 4134К |  |  |  | Самарин Д. В. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург

2024

**Часть 1.**

**Содержательная постановка задачи:**

Завод производит корпуса для холодильников пяти различных марок. Для их изготовления требуются трудозатраты, металл, пластик и краска. Каждому ресурсу соответствует определённый месячный лимит. Производство холодильников приносит различную прибыль в зависимости от марки. Необходимо составить такой план производства холодильников каждой марки, который максимизирует суммарную прибыль, учитывая ограничения по ресурсам.

**Переменные задачи:**

* x1​ — количество холодильников марки 1, которое нужно произвести.
* x2​ — количество холодильников марки 2, которое нужно произвести.
* x3​ — количество холодильников марки 3, которое нужно произвести.
* x4​ — количество холодильников марки 4, которое нужно произвести.
* x5​ — количество холодильников марки 5, которое нужно произвести.

**Целевая функция:**

Цель — максимизировать прибыль от производства холодильников. Формула прибыли будет суммой произведений количества произведённых холодильников каждой марки на прибыль от продажи одного холодильника этой марки:

Max Z=40x1​+70x2​+120x3​+120x4​+50x5​

**Ограничения:**

У нас есть четыре типа ресурсов, каждый из которых имеет свои ограничения по объему.

**Ограничение по трудозатратам (чел.-ч):**

Для каждой марки холодильников требуется определенное количество трудозатрат. Общие трудозатраты не должны превышать 900 человеко-часов:

2x1​+3x2​+5x3​+4x4​+4x5​≤900

**Ограничение по металлу (м²):**

Для производства холодильников требуется металл. Общий объём металла не должен превышать 8500 м²:

2x1​+2x2​+4x3​+5x4​+0x0≤8500

**Ограничение по пластику (м²):**

Общий объём используемого пластика не должен превышать 4000 м²:

1x1​+3x2​+2x3​+0x4​+4x4≤4000

**Ограничение по краске (кг):**

Для производства холодильников также требуется краска. Общий её объём не должен превышать 5000 кг:

1x1​+2x2​+3x3​+3x4​+2x5​≤5000

**Ограничения на переменные:**

Количество произведённых холодильников каждой марки должно быть неотрицательным, то есть:

x1​,x2​,x3​,x4​,x5​≥0

**Общая формулировка задачи ЛП:**

Целевая функция:

Max Z=40x1​+70x2​+120x3​+120x4​+50x5​

Ограничения:

2x1​+3x2​+5x3​+4x4​+4x5​≤900

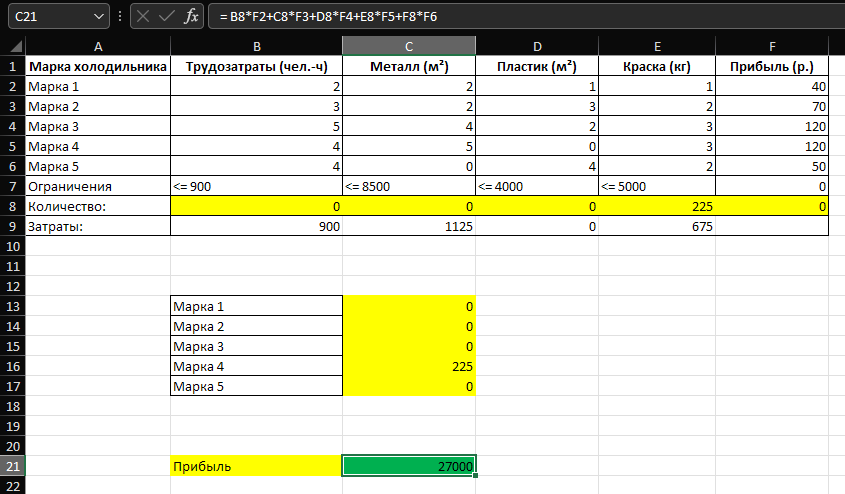
2x1​+2x2​+4x3​+5x4​+0x0≤8500

1x1​+3x2​+2x3​+0x4​+4x4≤4000

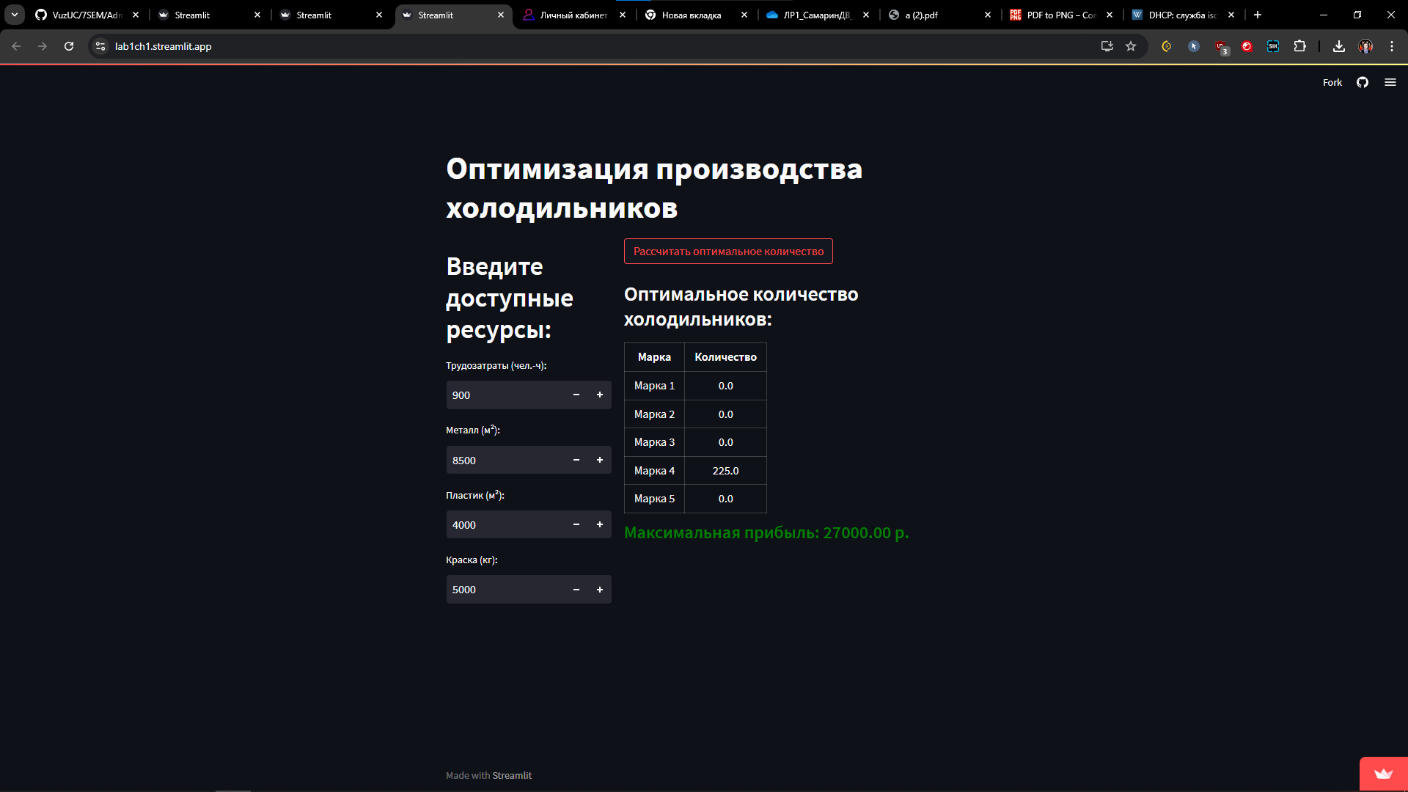
1x1​+2x2​+3x3​+3x4​+2x5​≤5000

x1​,x2​,x3​,x4​,x5​≥0

Ход работы в Excel:



Решение с помощью программирования:

Если хотите, то можете ознакомиться с написанной программой лично, перейдя по ссылке: <https://lab1ch1.streamlit.app/>

Рассмотрим код:

|  |
| --- |
| import streamlit as st  import pulp  # Функция для решения задачи оптимизации  def solve\_optimization\_problem(labor, metal, plastic, paint):  # Создаем задачу для максимизации прибыли  model = pulp.LpProblem("Maximize\_Profit", pulp.LpMaximize)  # Определяем переменные (целочисленные переменные x1, x2, x3, x4, x5)  x1 = pulp.LpVariable('x1', lowBound=0, cat='Integer')  x2 = pulp.LpVariable('x2', lowBound=0, cat='Integer')  x3 = pulp.LpVariable('x3', lowBound=0, cat='Integer')  x4 = pulp.LpVariable('x4', lowBound=0, cat='Integer')  x5 = pulp.LpVariable('x5', lowBound=0, cat='Integer')  # Целевая функция: максимизация прибыли  model += 40 \* x1 + 70 \* x2 + 120 \* x3 + 120 \* x4 + 50 \* x5, "Profit"  # Ограничения по ресурсам  model += 2 \* x1 + 3 \* x2 + 5 \* x3 + 4 \* x4 + 4 \* x5 <= labor # Трудозатраты  model += 2 \* x1 + 2 \* x2 + 4 \* x3 + 5 \* x4 + 0 \* x5 <= metal # Металл  model += 1 \* x1 + 3 \* x2 + 2 \* x3 + 0 \* x4 + 4 \* x5 <= plastic # Пластик  model += 1 \* x1 + 2 \* x2 + 3 \* x3 + 3 \* x4 + 2 \* x5 <= paint # Краска  # Решение задачи  model.solve()  # Возвращаем результаты  return {  "x1": x1.varValue,  "x2": x2.varValue,  "x3": x3.varValue,  "x4": x4.varValue,  "x5": x5.varValue,  "profit": pulp.value(model.objective)  }  # Streamlit интерфейс  st.title("Оптимизация производства холодильников")  # Создаем две колонки: одну для ввода данных, другую для отображения результатов  col1, col2 = st.columns([1, 2])  with col1:  st.header("Введите доступные ресурсы:")    # Поля для ввода ресурсов  labor = st.number\_input("Трудозатраты (чел.-ч):", min\_value=0, value=900, step=1)  metal = st.number\_input("Металл (м²):", min\_value=0, value=8500, step=1)  plastic = st.number\_input("Пластик (м²):", min\_value=0, value=4000, step=1)  paint = st.number\_input("Краска (кг):", min\_value=0, value=5000, step=1)  with col2:  if st.button("Рассчитать оптимальное количество"):  # Решаем задачу оптимизации  results = solve\_optimization\_problem(labor, metal, plastic, paint)  # Выводим результаты  st.subheader("Оптимальное количество холодильников:")  st.markdown(f"""  | \*\*Марка\*\* | \*\*Количество\*\* |  |:---------:|:--------------:|  | Марка 1 | {results['x1']} |  | Марка 2 | {results['x2']} |  | Марка 3 | {results['x3']} |  | Марка 4 | {results['x4']} |  | Марка 5 | {results['x5']} |  """)  st.markdown(f"<h4 style='color:green;'>Максимальная прибыль: {results['profit']:.2f} р.</h4>", unsafe\_allow\_html=True) |

**Часть 2.**

**Цель работы:**

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

**Ход работы:**

1. Ознакомиться со справочными сведениями;

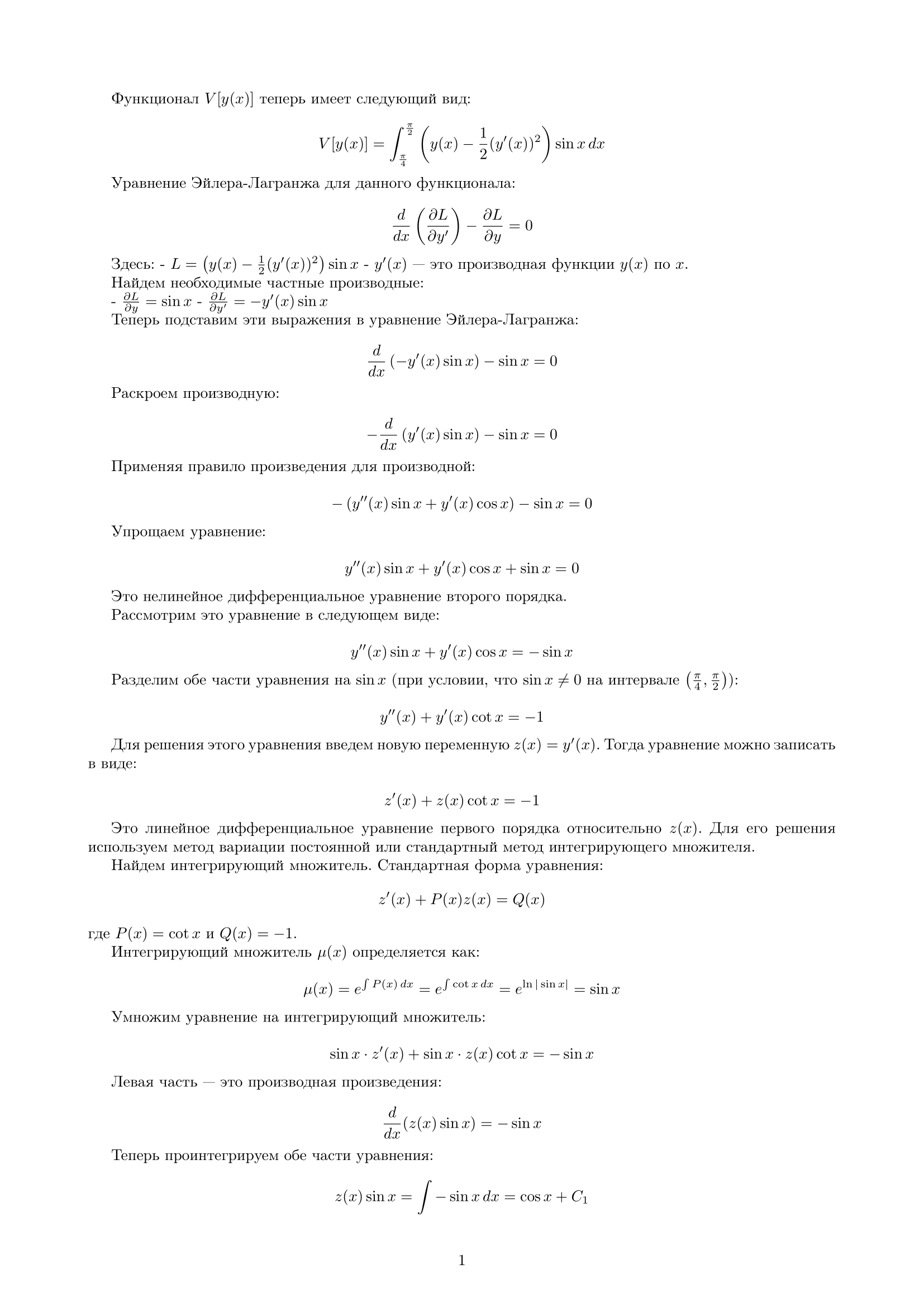
2. Записать и решить уравнение Эйлера-Лагранжа для оптимизационного функционала.

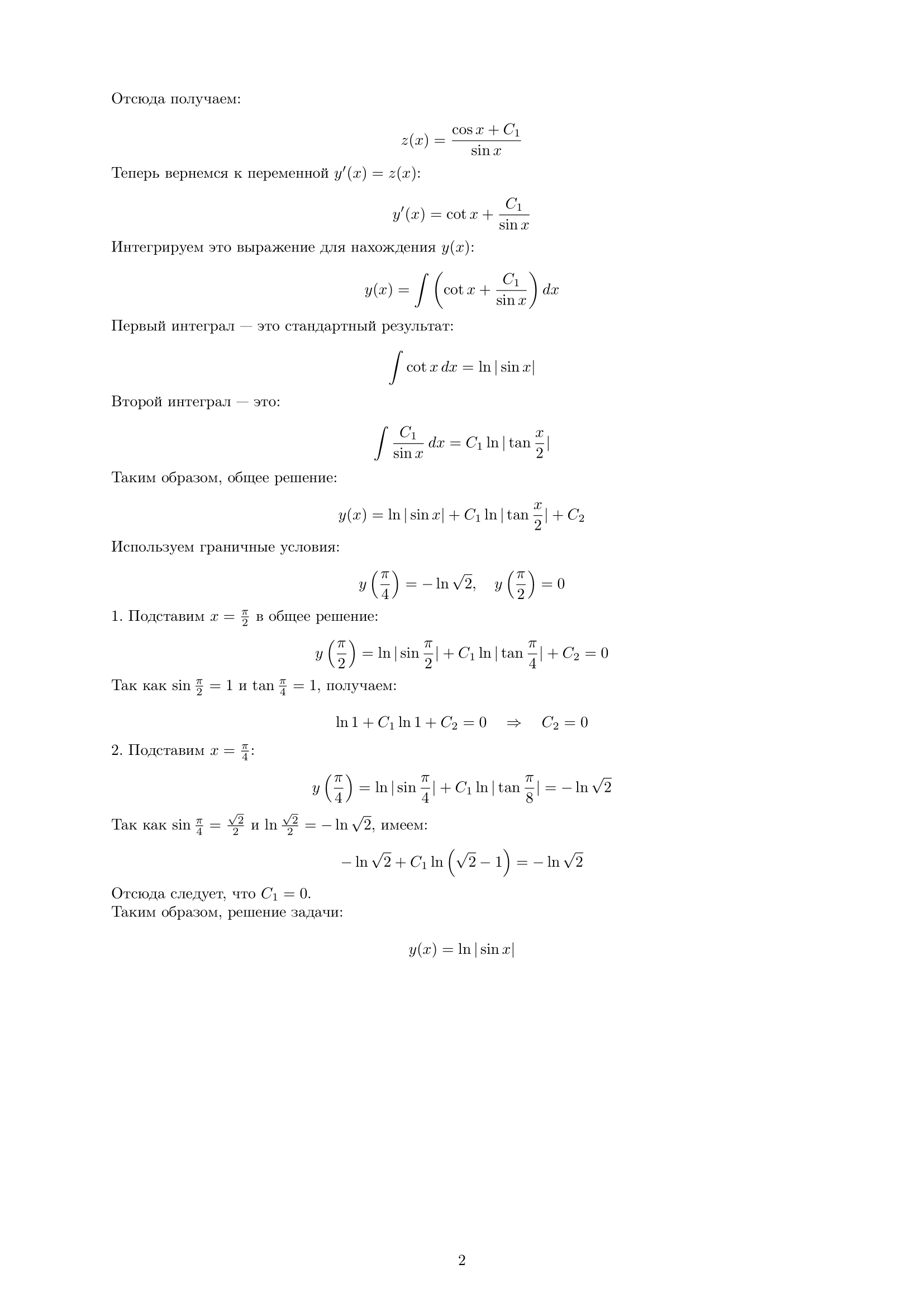
3. Разработать программу, моделирующую алгоритм поиска оптимального решения для формализованной задачи, используя вычислительный пакет MatLab и/или язык программирования Python. За аналитическое решение («в ручную») ДУ – дополнительные баллы-бонусы.

4. Подготовить и устно защитить отчет о работе.

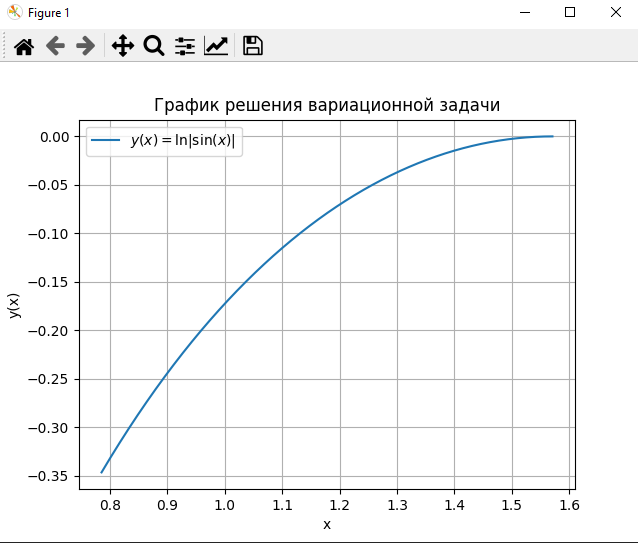
**Рассмотрим полученное решение.**

**Аналитическое:**



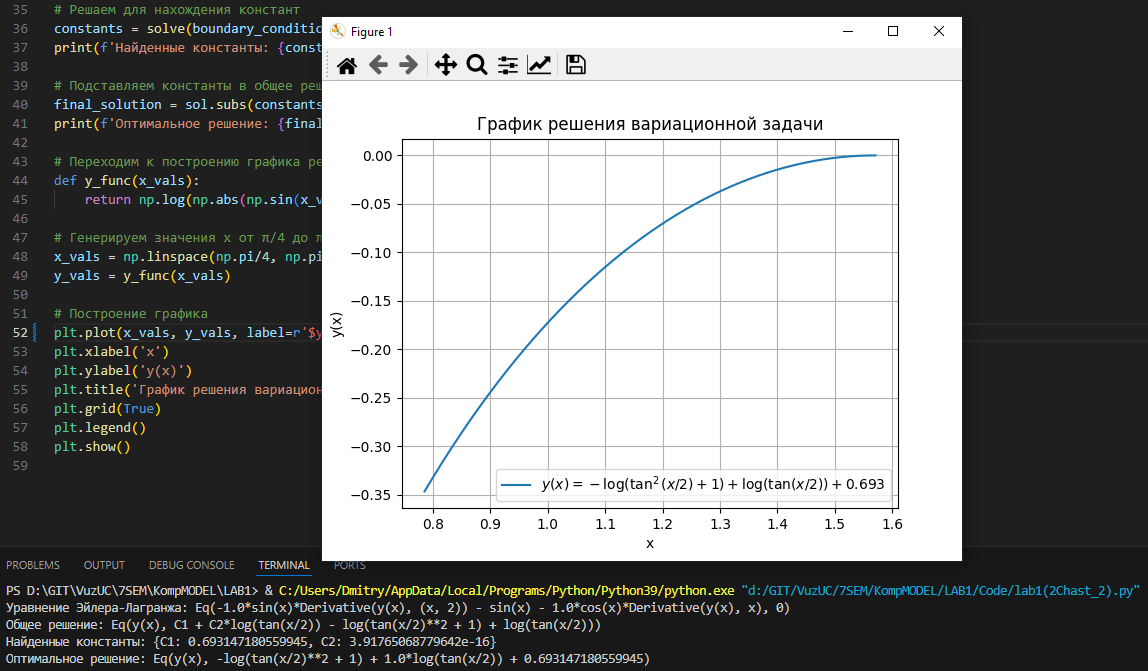


Полученный график:



**Решение с помощью PY.**

Метод - численный, с выводом графика:



Решение близко к ожидаемому y(x) = ln | sin x |. Стоит понимать, что численные методы как правило дают эквивалентные, но не идентичные по форме выражения. Поэтому, чтобы убедиться в правильности хода решений мы строим графики функций.

Таким образом, решением функции является y(x) = ln | sin x |**.**

Предлагаю ознакомиться с кодом программы:

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  import streamlit as st  from sympy import symbols, Function, Derivative, dsolve, Eq, sin, log, pi, solve  # Определяем символы  x = symbols('x')  y = Function('y')(x)  dy = Derivative(y, x)  # Определяем константы C1 и C2  C1, C2 = symbols('C1 C2')  # Определяем лагранжиан (L)  L = (y - (1/2) \* dy\*\*2) \* sin(x)  # Вычисляем частные производные  dL\_dy = L.diff(y) # Частная производная L по y  dL\_ddy = L.diff(dy) # Частная производная L по dy  d\_dx\_dL\_ddy = Derivative(dL\_ddy, x).doit() # Производная по x от dL\_ddy  # Уравнение Эйлера-Лагранжа  euler\_lagrange\_eq = Eq(d\_dx\_dL\_ddy - dL\_dy, 0)  st.write(f'Уравнение Эйлера-Лагранжа: {euler\_lagrange\_eq}')  # Решаем уравнение Эйлера-Лагранжа  sol = dsolve(euler\_lagrange\_eq)  st.write(f'Общее решение: {sol}')  # Задаем граничные условия: y(π/4) = -ln(√2) и y(π/2) = 0  boundary\_conditions = [  Eq(sol.rhs.subs(x, pi/4), -log(np.sqrt(2))), # y(π/4) = -ln(√2)  Eq(sol.rhs.subs(x, pi/2), 0) # y(π/2) = 0  ]  # Решаем для нахождения констант  constants = solve(boundary\_conditions, (C1, C2))  st.write(f'Найденные константы: {constants}')  # Подставляем константы в общее решение  final\_solution = sol.subs(constants)  st.write(f'Оптимальное решение: {final\_solution}')  # Переходим к построению графика решения y(x) = ln|sin(x)|  def y\_func(x\_vals):  return np.log(np.abs(np.sin(x\_vals)))  # Генерируем значения x от π/4 до π/2  x\_vals = np.linspace(np.pi/4, np.pi/2, 500)  y\_vals = y\_func(x\_vals)  # Построение графика  plt.plot(x\_vals, y\_vals, label=r'$y(x) = \ln|\sin(x)|$')  plt.xlabel('x')  plt.ylabel('y(x)')  plt.title('График решения вариационной задачи')  plt.grid(True)  plt.legend()  # Отображение графика в Streamlit  st.pyplot(plt) |

**Вывод:**

Выполнив лабораторную работу №1 (1 и 2 часть), мы освоили средства моделирования задач нелинейного программирования и смогли решить вариационную задачу.