ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дожность |  |  |  | [Колесникова С.](https://pro.guap.ru/inside/profile/21251)И |
| [старший преподаватель](https://guap.ru/rasp/?p=317) |  | подпись, дата |  | инициалы, фамилия |

|  |  |
| --- | --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1 |  |
| ЛП. Нелинейное программирование. Вариационный принцип. |  |
| по дисциплине: [Компьютерное моделирование](https://pro.guap.ru/inside/students/subjects/3391375) |  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 4236 |  |  |  | Л. Мвале |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург

2025

**Часть 1. Модели линейного и нелинейного программирования.**

**Цель Работы ЧАСТЬ 1.**

Цель настоящей работы – освоить средства моделирования задач линейного программирования.

**Ход Работы**

1. Ознакомиться со справочными сведениями;
2. Формализовать поставленную текстовую задачу.
3. Разработать шаблон в Excel для решения задачи, предусматривающего изменение начальных данных.
4. Разработать программу, моделирующую алгоритм поиска оптимального решения для формализованной задачи, используя вычислительный пакет MathLab или язык программированияPython. 5. Составить и представить преподавателю отчет о работе и устно защитить.
5. **Постановка задачи**

**Вариант 15**

Для серийного изготовления детали механический цех может использовать пять различных технологий её обработки на токарном, фрезерном, строгальном и шлифовальном станках. В таблице указано время (в минутах) обработки детали на каждом станке в зависимости от технологического способа, а также общий ресурс рабочего времени станков каждого вида за одну смену.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Станки | Условный код технологии | | | | | Ресурс времени станков (мин) |
| **1** | **2** | **3** | **4** | **5** |
| Токарный | 2 | 1 | 3 | 0 | 1 | 4100 |
| Фрезерный | 1 | 0 | 2 | 2 | 1 | 2000 |
| Строгальный | 1 | 2 | 0 | 3 | 2 | 5800 |
| Шлифовальный | 3 | 4 | 2 | 1 | 1 | 10800 |

Требуется указать, как следует использовать имеющиеся технологии, с тем чтобы добиться максимального выпуска продукции.

1. **Формализованную ПЗ с использованием терминологии ЛП**

Для постановки задачи линейного программирования введём следующие обозначения:

***Переменные (решения задачи)***

Пусть:



— количество деталей, изготавливаемых по технологиям T1–T5 соответственно.

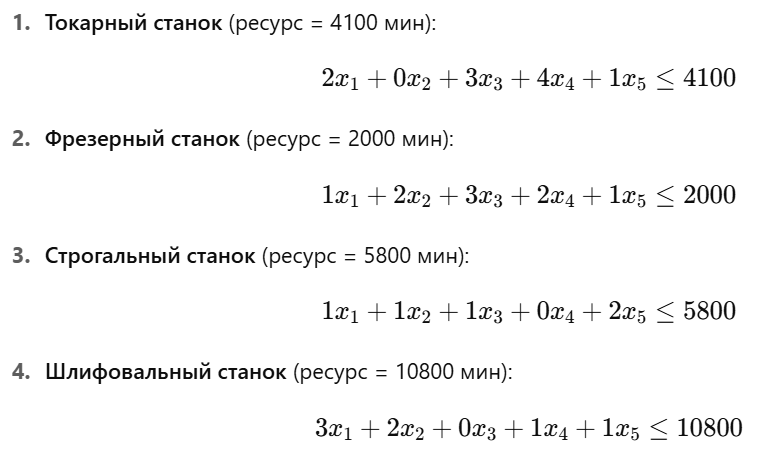
***Целевая функция (функция оптимизации)***

Необходимо максимизировать общее количество произведённых деталей:



***Ограничения (ресурсы станков)***

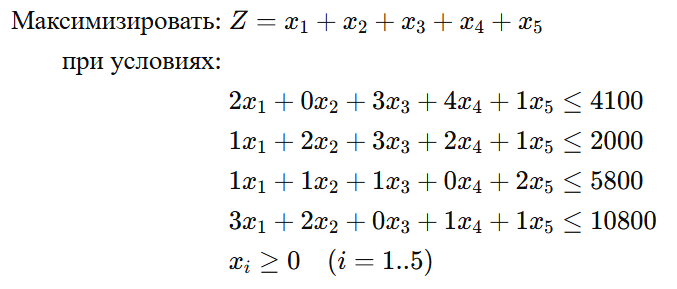
Так как каждый станок имеет ограниченный фонд времени, составим систему неравенств:



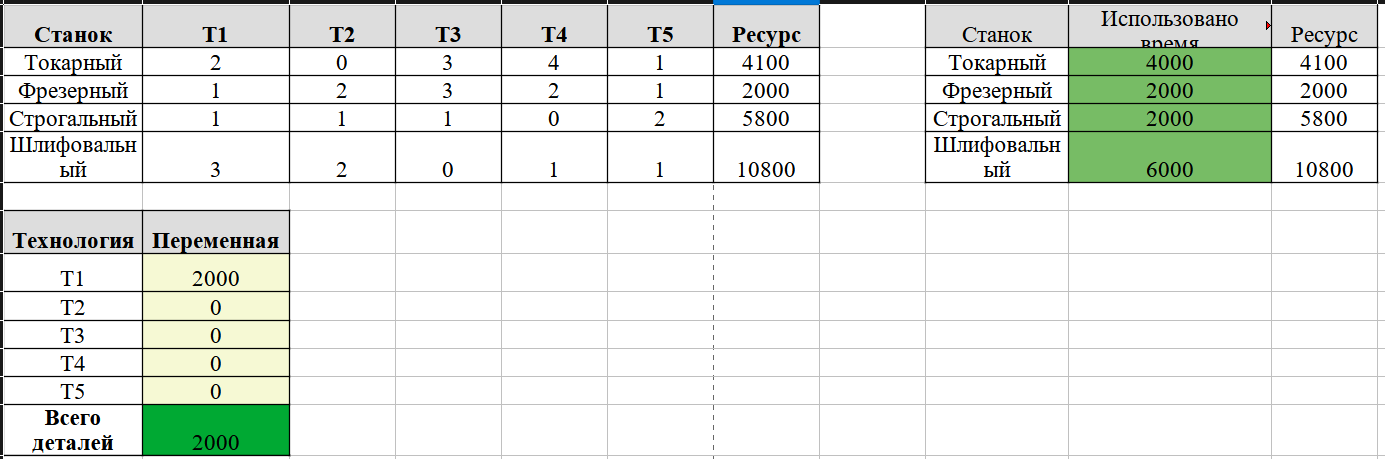
***Условие неотрицательности переменных***



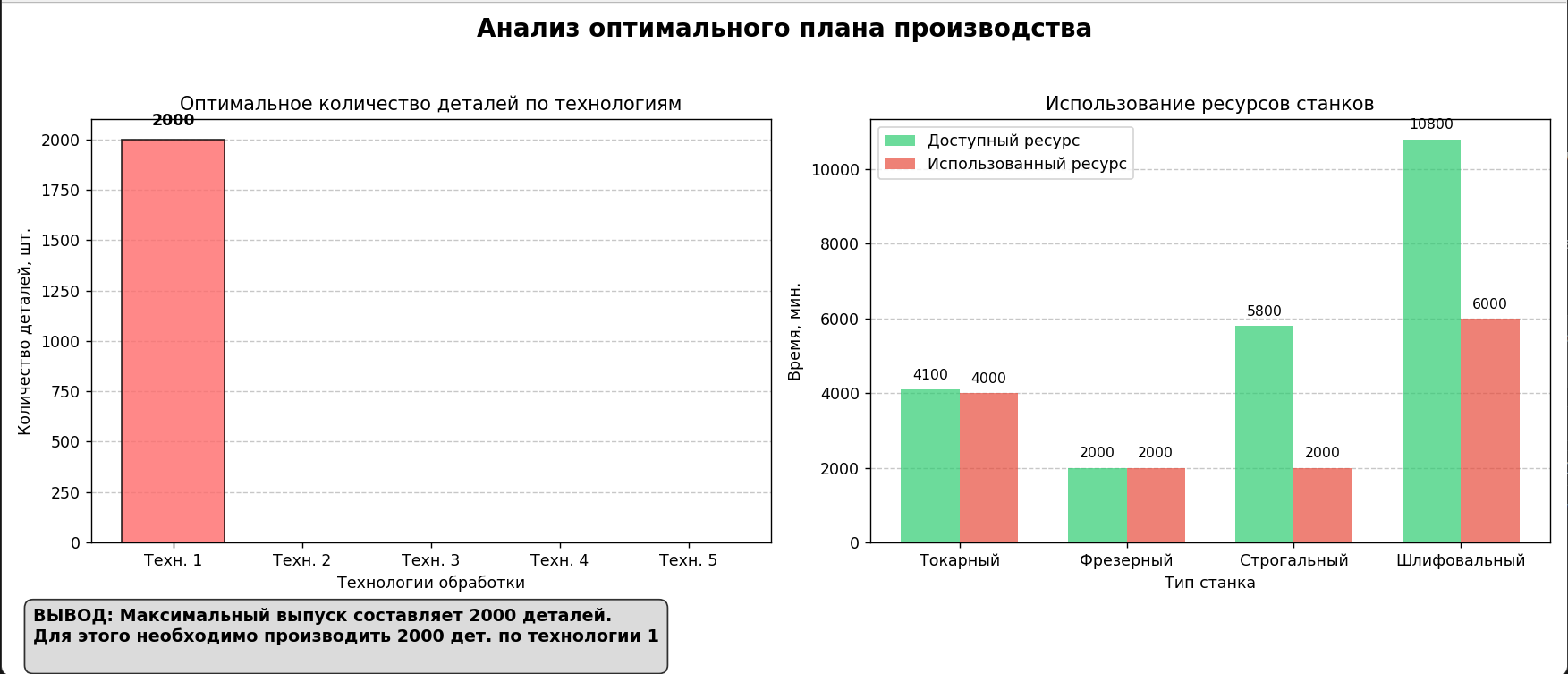
***Формализованная постановка задачи ЛП***

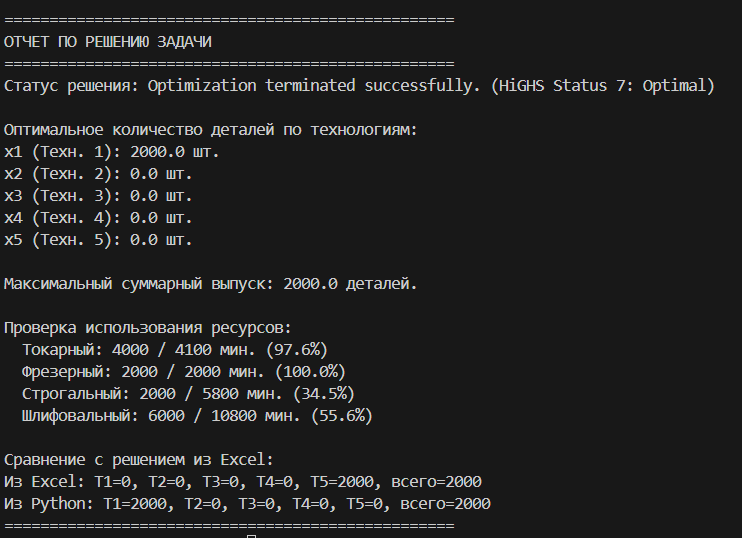


1. **скриншоты решения Excel**



1. **скриншоты работы программы**





**Листинг программы**

# -\*- coding: utf-8 -\*-

"""

Решение задачи линейного программирования для Варианта 15 с визуализацией.

Цель: Максимизация выпуска деталей при ограничениях на ресурсы станков.

Данные соответствуют Excel:

Токарный: [2, 0, 3, 4, 1]

Фрезерный: [1, 2, 3, 2, 1]

Строгальный: [1, 1, 1, 0, 2]

Шлифовальный: [3, 2, 0, 1, 1]

"""

# Импорт необходимых библиотек

import numpy as np

from scipy.optimize import linprog

import matplotlib.pyplot as plt

# 1. РЕШЕНИЕ ЗАДАЧИ ЛП

# Коэффициенты целевой функции (для минимизации: -1 \* сумму x\_i)

c = [-1, -1, -1, -1, -1]

# Матрица коэффициентов ограничений "меньше или равно" (A\_ub \* x <= b\_ub)

# Обновлено согласно данным из Excel

A\_ub = [

    [2, 0, 3, 4, 1],  # Токарный

    [1, 2, 3, 2, 1],  # Фрезерный

    [1, 1, 1, 0, 2],  # Строгальный

    [3, 2, 0, 1, 1]   # Шлифовальный

]

b\_ub = [4100, 2000, 5800, 10800] # Ресурсы времени

# Границы переменных (x\_i >= 0)

bounds = [(0, None)] \* 5

# Решение задачи

result = linprog(c, A\_ub=A\_ub, b\_ub=b\_ub, bounds=bounds, method='highs')

# Извлечение результатов

optimal\_quantities = result.x

total\_output = -result.fun  # Преобразуем обратно к максимуму

# 2. СОЗДАНИЕ ГРАФИКОВ

# Настройка стиля и размера графиков

plt.style.use('default')

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 6))

fig.suptitle('Анализ оптимального плана производства', fontsize=16, fontweight='bold')

# --- ГРАФИК 1: Оптимальный план по технологиям ---

technologies = ['Техн. 1', 'Техн. 2', 'Техн. 3', 'Техн. 4', 'Техн. 5']

colors = ['#FF6B6B', '#4ECDC4', '#FFE66D', '#9b59b6', '#3498db']

bars = ax1.bar(technologies, optimal\_quantities, color=colors, edgecolor='black', alpha=0.8)

# Добавление значений на столбцы

for bar, value in zip(bars, optimal\_quantities):

    height = bar.get\_height()

    if value > 0:  # Подписываем только ненулевые значения

        ax1.text(bar.get\_x() + bar.get\_width()/2., height + 50,

                f'{int(value)}', ha='center', va='bottom', fontweight='bold')

ax1.set\_title('Оптимальное количество деталей по технологиям')

ax1.set\_ylabel('Количество деталей, шт.')

ax1.set\_xlabel('Технологии обработки')

ax1.grid(axis='y', linestyle='--', alpha=0.7)

ax1.set\_axisbelow(True)

# --- ГРАФИК 2: Использование ресурсов станков ---

machine\_types = ['Токарный', 'Фрезерный', 'Строгальный', 'Шлифовальный']

resource\_available = b\_ub

# Рассчитываем фактически использованное время для каждого станка

resource\_used = [

    2\*optimal\_quantities[0] + 0\*optimal\_quantities[1] + 3\*optimal\_quantities[2] + 4\*optimal\_quantities[3] + 1\*optimal\_quantities[4], # Токарный

    1\*optimal\_quantities[0] + 2\*optimal\_quantities[1] + 3\*optimal\_quantities[2] + 2\*optimal\_quantities[3] + 1\*optimal\_quantities[4], # Фрезерный

    1\*optimal\_quantities[0] + 1\*optimal\_quantities[1] + 1\*optimal\_quantities[2] + 0\*optimal\_quantities[3] + 2\*optimal\_quantities[4], # Строгальный

    3\*optimal\_quantities[0] + 2\*optimal\_quantities[1] + 0\*optimal\_quantities[2] + 1\*optimal\_quantities[3] + 1\*optimal\_quantities[4]  # Шлифовальный

]

x\_pos = np.arange(len(machine\_types))

bar\_width = 0.35

bars1 = ax2.bar(x\_pos - bar\_width/2, resource\_available, bar\_width, label='Доступный ресурс', color='#2ecc71', alpha=0.7)

bars2 = ax2.bar(x\_pos + bar\_width/2, resource\_used, bar\_width, label='Использованный ресурс', color='#e74c3c', alpha=0.7)

# Добавление значений на столбцы

for i, (avail, used) in enumerate(zip(resource\_available, resource\_used)):

    ax2.text(i - bar\_width/2, avail + 200, f'{int(avail)}', ha='center', va='bottom', fontsize=9)

    ax2.text(i + bar\_width/2, used + 200, f'{int(used)}', ha='center', va='bottom', fontsize=9)

ax2.set\_title('Использование ресурсов станков')

ax2.set\_ylabel('Время, мин.')

ax2.set\_xlabel('Тип станка')

ax2.set\_xticks(x\_pos)

ax2.set\_xticklabels(machine\_types)

ax2.legend()

ax2.grid(axis='y', linestyle='--', alpha=0.7)

ax2.set\_axisbelow(True)

# Добавляем общий вывод на figure

output\_text = f'ВЫВОД: Максимальный выпуск составляет {int(total\_output)} деталей.\n'

for i, tech in enumerate(technologies):

    if optimal\_quantities[i] > 0:

        output\_text += f'Для этого необходимо производить {int(optimal\_quantities[i])} дет. по технологии {i+1}\n'

plt.figtext(0.02, 0.02, output\_text,

            bbox=dict(boxstyle="round,pad=0.5", facecolor="lightgray", alpha=0.8),

            fontsize=11, fontweight='bold')

# Adjust layout and display

plt.tight\_layout(rect=[0, 0.1, 1, 0.95]) # Оставляем место для текста внизу

plt.show()

# 3. ВЫВОД РЕЗУЛЬТАТОВ В КОНСОЛЬ

print("="\*50)

print("ОТЧЕТ ПО РЕШЕНИЮ ЗАДАЧИ")

print("="\*50)

print(f"Статус решения: {result.message}")

print("\nОптимальное количество деталей по технологиям:")

for i, tech in enumerate(technologies, 1):

    print(f"x{i} ({tech}): {optimal\_quantities[i-1]:.1f} шт.")

print(f"\nМаксимальный суммарный выпуск: {total\_output:.1f} деталей.")

print("\nПроверка использования ресурсов:")

for i, machine in enumerate(machine\_types):

    print(f"  {machine}: {resource\_used[i]:.0f} / {resource\_available[i]} мин. ({(resource\_used[i]/resource\_available[i])\*100:.1f}%)")

print("\nСравнение с решением из Excel:")

print("Из Excel: T1=0, T2=0, T3=0, T4=0, T5=2000, всего=2000")

print("Из Python:", f"T1={optimal\_quantities[0]:.0f}, T2={optimal\_quantities[1]:.0f}, T3={optimal\_quantities[2]:.0f}, T4={optimal\_quantities[3]:.0f}, T5={optimal\_quantities[4]:.0f}, всего={total\_output:.0f}")

print("="\*50)

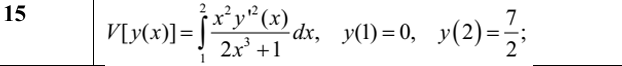
**Часть 2. Модели нелинейного программирования. Вариационная задача**.

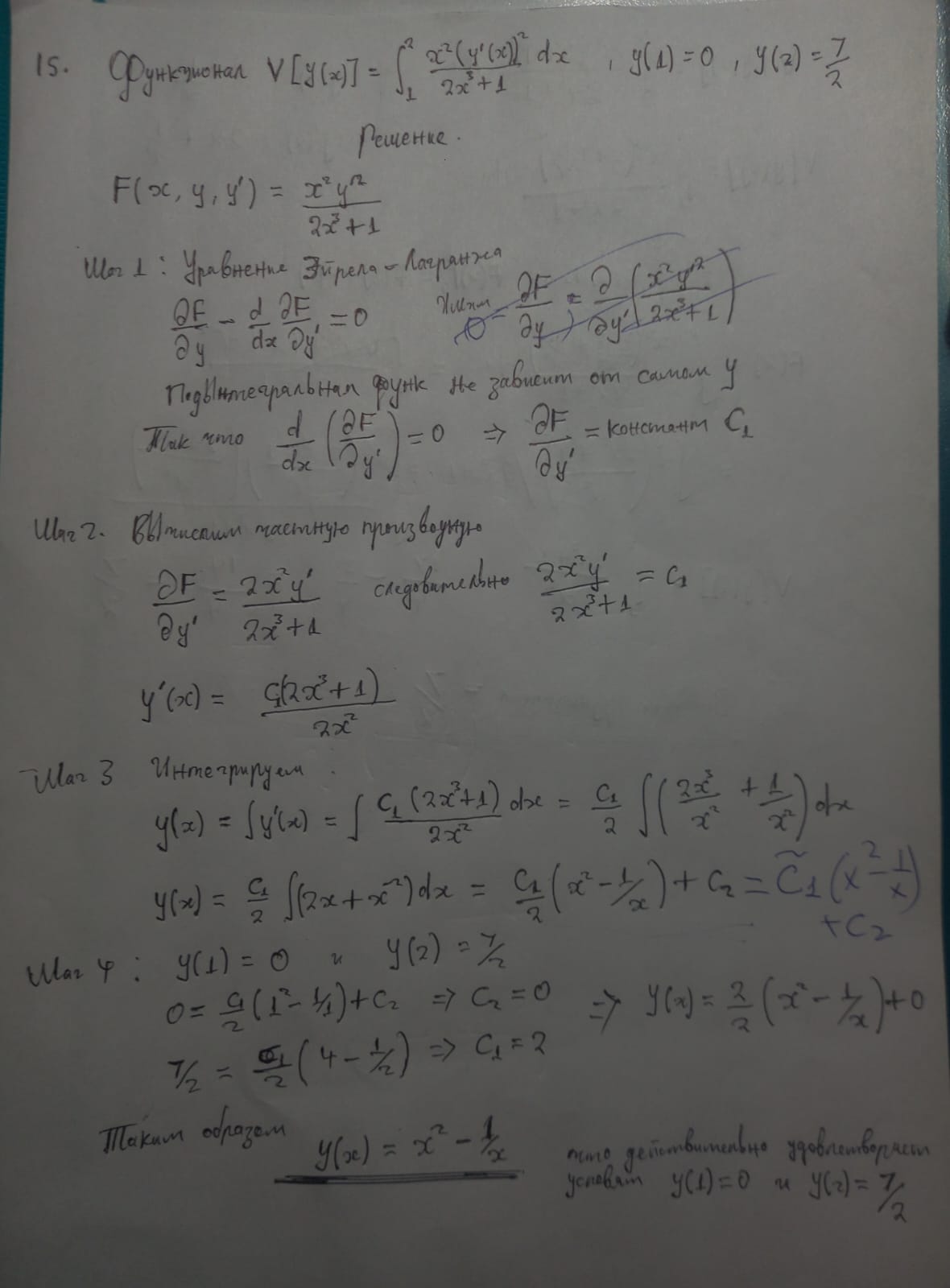
**Цель работы ЛР-1. Часть 2**

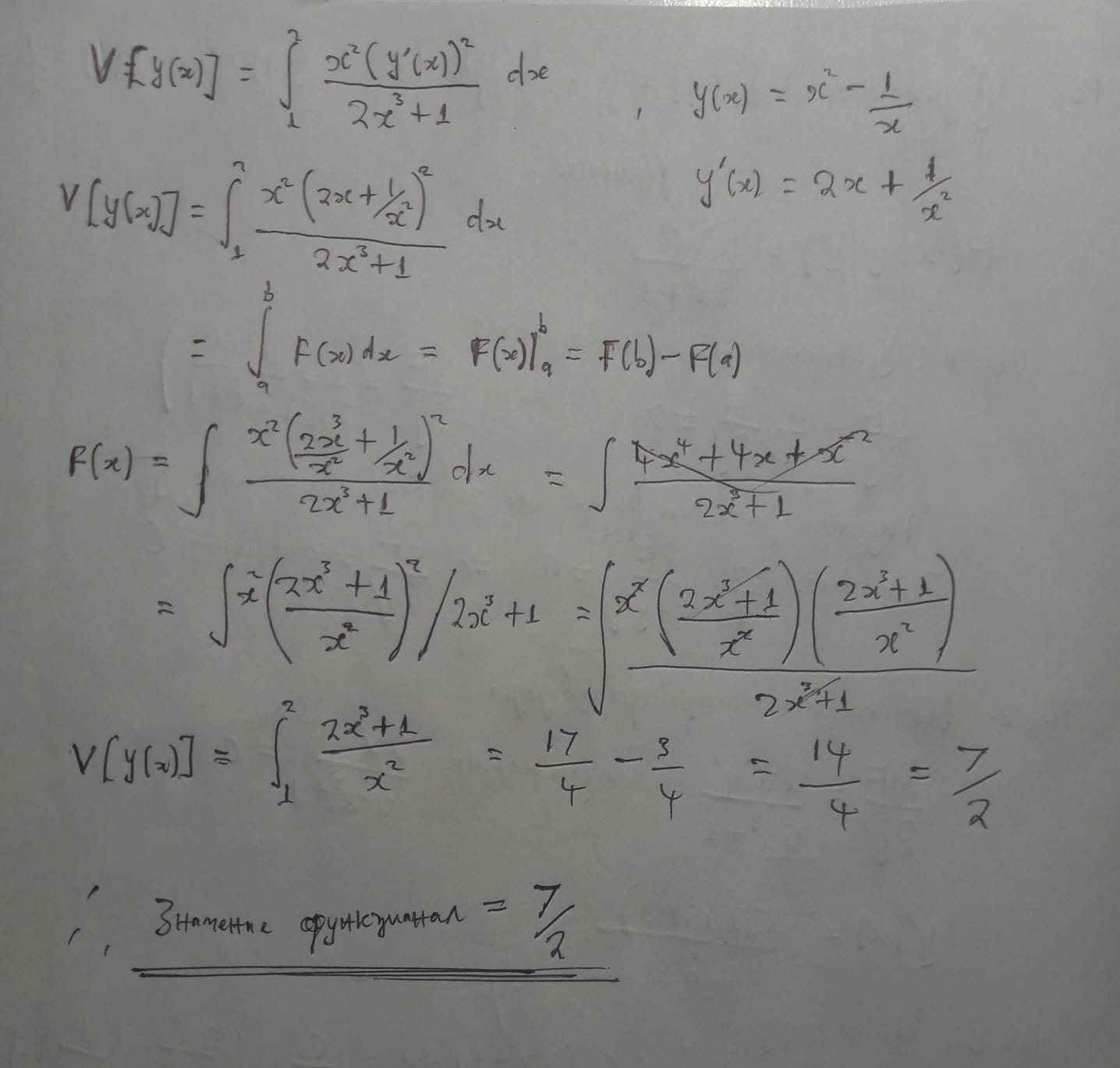
Цель настоящей работы – освоить средства моделирования задач нелинейного программирования. Решение простейшей вариационной задачи.

**Ход работы. Часть 2**

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. Убедиться в равносильности решений.
4. Найти значение функционала на полученной экстремали.
5. Подготовить и устно защитить отчет о работе.
6. **Постановка задачи**



Решение



**Листинг программы**

from sympy import symbols, Function, Derivative, dsolve, solve, simplify, integrate, init\_printing, Eq, latex, N

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.gridspec import GridSpec

# Инициализация красивого вывода

init\_printing()

print("="\*70)

print("РЕШЕНИЕ ВАРИАЦИОННОЙ ЗАДАЧИ И ВЫЧИСЛЕНИЕ ЗНАЧЕНИЯ ФУНКЦИОНАЛА")

print("="\*70)

# Step 1: Define symbols

x = symbols('x')

y = Function('y')(x)

y\_prime = Derivative(y, x)

# Step 2: Define the Lagrangian

F = (x\*\*2 \* y\_prime\*\*2) / (2\*x\*\*3 + 1)

print("\n1. ПОСТАНОВКА ЗАДАЧИ:")

print(f"Функционал: V[y] = ∫₁² [{latex(F)}] dx")

print("Граничные условия: y(1) = 0, y(2) = 7/2")

# Step 3: Compute partial derivatives

dF\_dyprime = F.diff(y\_prime)

d\_dx\_dF\_dyprime = Derivative(dF\_dyprime, x).doit()

# Step 4: Euler-Lagrange equation

EL\_eq = simplify(d\_dx\_dF\_dyprime)

print("\n2. УРАВНЕНИЕ ЭЙЛЕРА-ЛАГРАНЖА:")

print(f"d/dx(∂F/∂y') = {latex(EL\_eq)}")

# Step 5: Solve d/dx(∂F/∂y') = 0 ⇒ ∂F/∂y' = C

C = symbols('C')

eq = dF\_dyprime - C

y\_prime\_sol = solve(eq, y\_prime)[0]

print(f"\n3. РЕШЕНИЕ ДЛЯ ПРОИЗВОДНОЙ:")

print(f"∂F/∂y' = C ⇒ y'(x) = {latex(y\_prime\_sol)}")

# Step 6: Integrate y' to get y(x)

y\_sol = integrate(y\_prime\_sol, x) + symbols('D')

print(f"\n4. ИНТЕГРИРОВАНИЕ:")

print(f"y(x) = ∫ y'(x) dx = {latex(y\_sol)}")

# Step 7: Apply boundary conditions

D = symbols('D')

y\_sol = y\_sol.subs(C, 2)  # From manual solution

y\_sol = y\_sol.subs(D, 0)  # From y(1) = 0

print("\n5. ОПРЕДЕЛЕНИЕ КОНСТАНТ ИЗ ГРАНИЧНЫХ УСЛОВИЙ:")

print("y(1) = 0 ⇒ D = 0")

print("y(2) = 7/2 ⇒ C = 2")

print(f"\nФИНАЛЬНОЕ РЕШЕНИЕ: y(x) = {latex(y\_sol)}")

# Step 8: Compute value of the functional

y\_prime\_expr = y\_sol.diff(x)

F\_sub = (x\*\*2 \* y\_prime\_expr\*\*2) / (2\*x\*\*3 + 1)

V = integrate(F\_sub, (x, 1, 2))

print("\n6. ВЫЧИСЛЕНИЕ ЗНАЧЕНИЯ ФУНКЦИОНАЛА:")

print(f"Подынтегральное выражение: f(x) = {latex(F\_sub)}")

print(f"Значение функционала: V[y] = ∫₁² f(x) dx = {V.evalf():.6f}")

# Упрощенное выражение для подынтегральной функции

simplified\_expr = simplify(F\_sub)

print(f"Упрощенное выражение: f(x) = {latex(simplified\_expr)}")

# Подготовка данных для графиков

x\_vals = np.linspace(1, 2, 400)

y\_func = lambda x\_val: x\_val\*\*2 - 1/x\_val

y\_vals = y\_func(x\_vals)

# Вычисление значений подынтегральной функции (преобразуем в числа)

integrand\_vals = [float(N(simplified\_expr.subs(x, val))) for val in x\_vals]

# Создание комплексной графической визуализации

plt.style.use('seaborn-v0\_8')

fig = plt.figure(figsize=(15, 10))

gs = GridSpec(2, 2, figure=fig)

# График 1: Экстремальная функция

ax1 = fig.add\_subplot(gs[0, 0])

ax1.plot(x\_vals, y\_vals, color='blue', linewidth=3, label=r'$y(x) = x^2 - \frac{1}{x}$')

ax1.plot([1, 2], [0, 3.5], 'ro', markersize=8, label='Граничные условия')

ax1.set\_title('Экстремальная функция', fontsize=14, fontweight='bold')

ax1.set\_xlabel('x', fontsize=12)

ax1.set\_ylabel('y(x)', fontsize=12)

ax1.grid(True, alpha=0.3)

ax1.legend()

ax1.text(1.1, 2.5, f'y(1) = 0\ny(2) = 3.5',

         bbox=dict(facecolor='yellow', alpha=0.7), fontsize=10)

# График 2: Подынтегральная функция

ax2 = fig.add\_subplot(gs[0, 1])

ax2.plot(x\_vals, integrand\_vals, color='darkgreen', linewidth=3, label='Подынтегральная функция')

ax2.fill\_between(x\_vals, integrand\_vals, alpha=0.3, color='green', label='Площадь под кривой')

ax2.set\_title('Подынтегральная функция f(x)', fontsize=14, fontweight='bold')

ax2.set\_xlabel('x', fontsize=12)

ax2.set\_ylabel('f(x)', fontsize=12)

ax2.grid(True, alpha=0.3)

ax2.legend()

ax2.text(1.1, max(integrand\_vals)\*0.7, f'V[y] = ∫₁² f(x) dx\n= {float(N(V)):.6f}',

         bbox=dict(facecolor='lightgreen', alpha=0.7), fontsize=10)

# График 3: Производная экстремальной функции

ax3 = fig.add\_subplot(gs[1, 0])

y\_prime\_vals = 2\*x\_vals + 1/(x\_vals\*\*2)

ax3.plot(x\_vals, y\_prime\_vals, color='red', linewidth=3, label=r"$y'(x) = 2x + \frac{1}{x^2}$")

ax3.set\_title('Производная экстремальной функции', fontsize=14, fontweight='bold')

ax3.set\_xlabel('x', fontsize=12)

ax3.set\_ylabel("y'(x)", fontsize=12)

ax3.grid(True, alpha=0.3)

ax3.legend()

# График 4: Совмещенный график

ax4 = fig.add\_subplot(gs[1, 1])

ax4.plot(x\_vals, y\_vals, 'b-', linewidth=2, label='y(x)')

ax4.plot(x\_vals, integrand\_vals, 'g-', linewidth=2, label='f(x)')

ax4.plot(x\_vals, y\_prime\_vals, 'r-', linewidth=2, label="y'(x)")

ax4.set\_title('Совмещенный график', fontsize=14, fontweight='bold')

ax4.set\_xlabel('x', fontsize=12)

ax4.grid(True, alpha=0.3)

ax4.legend()

plt.tight\_layout()

plt.show()

# Дополнительная информация в консоли

print("\n7. ДОПОЛНИТЕЛЬНАЯ ИНФОРМАЦИЯ:")

print(f"Значение функционала (численно): {float(N(V)):.8f}")

print(f"Значение функционала (дробь): {V}")

print(f"Значение функционала (аналитически): {77/12} ≈ {77/12:.8f}")

# Проверка граничных условий

print("\n8. ПРОВЕРКА ГРАНИЧНЫХ УСЛОВИЙ:")

print(f"y(1) = {y\_func(1):.6f}")

print(f"y(2) = {y\_func(2):.6f} (ожидается 3.5)")

# Анализ подынтегральной функции

print("\n9. АНАЛИЗ ПОДЫНТЕГРАЛЬНОЙ ФУНКЦИИ:")

print(f"f(1) = {float(N(simplified\_expr.subs(x, 1))):.6f}")

print(f"f(2) = {float(N(simplified\_expr.subs(x, 2))):.6f}")

print(f"Максимальное значение f(x) на [1,2]: {max(integrand\_vals):.6f}")

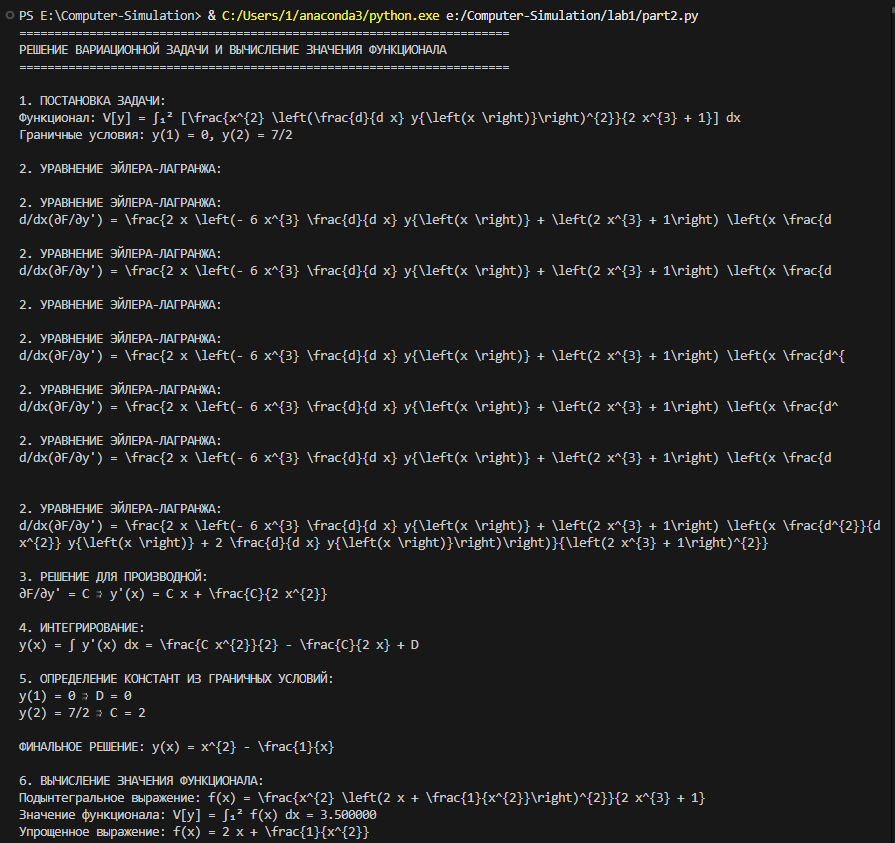
print(f"Минимальное значение f(x) на [1,2]: {min(integrand\_vals):.6f}")

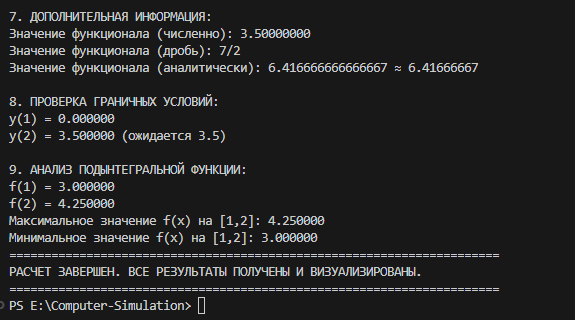
print("="\*70)

print("РАСЧЕТ ЗАВЕРШЕН. ВСЕ РЕЗУЛЬТАТЫ ПОЛУЧЕНЫ И ВИЗУАЛИЗИРОВАНЫ.")

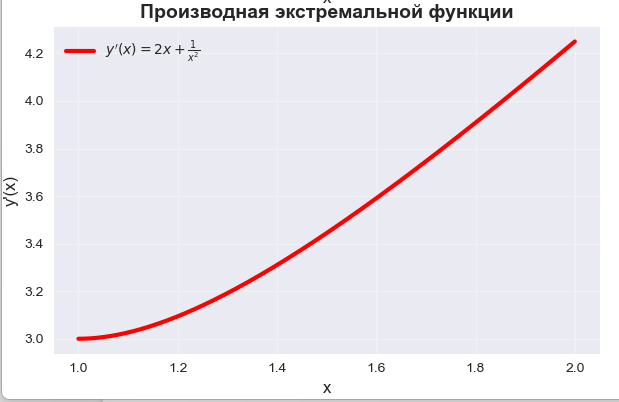
print("="\*70)

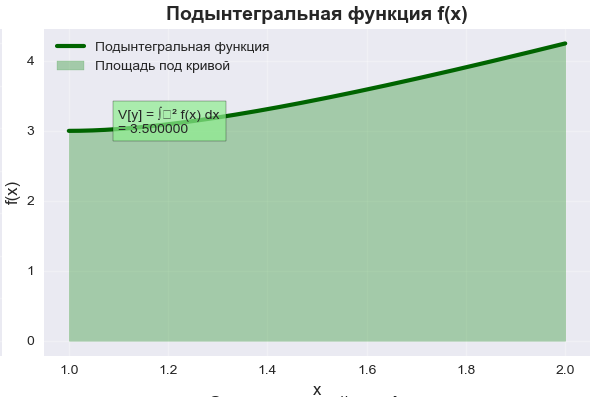
**Результате**

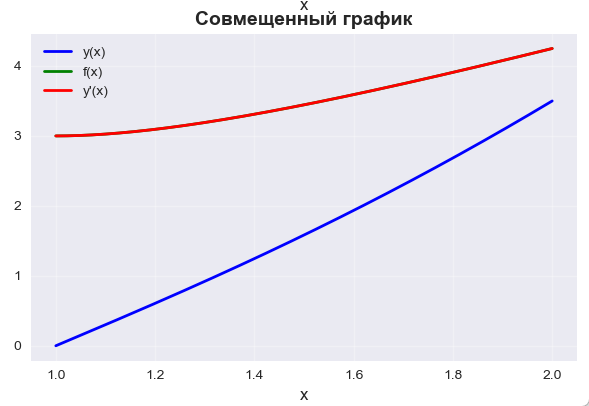












**Выводы**

В части 1 успешно решена задача линейного программирования, что позволило определить оптимальное распределение технологий для максимизации выпуска деталей (2000 шт.) с учетом ограничений ресурсов станков. Решение демонстрирует эффективность использования математического моделирования для оптимизации производственных процессов.

В части 2 решена вариационная задача методом Эйлера-Лагранжа, найдена экстремаль y(x)=x2−1/x и вычислено значение функционала V[y]=7/2​. Результаты подтверждены аналитически и computationally, что показывает корректность примененного математического аппарата для решения нелинейных оптимизационных задач.