Министерство науки и образования РФ

Федеральное государственное бюджетное учреждение

высшего образования

**«Тверской государственный технический университет»**

(ТвГТУ)

Кафедра программного обеспечения

**Отчет по лабораторной работе №1**

по дисциплине: «Компьютерные системы моделирования»

Тема: «Построение аналитической модели по текстовому описанию задачи и по результатам экспериментов»

|  |
| --- |
| Выполнил:  студент группы  Б.ПИН.РИС-22.06  Борздов М.А. |
| Проверила:  старший преподаватель кафедры ПО  Корнеева Е.И. |

Тверь 2025

# **Теория по варианту**

Основное требование к модели — её **адекватность**. То есть соответствие модели оригиналу, результаты моделирования подтверждаются и могут служить основой для прогнозирования процессов.  
Оно также включает:

**Точность** – степень точности копирования свойств объекта моделью.

**Цель**– для решения какой задачи создается модель.

**Управляемость**– например, число параметров модели.

**Целостность** – степень слаженности подсистеме модели.

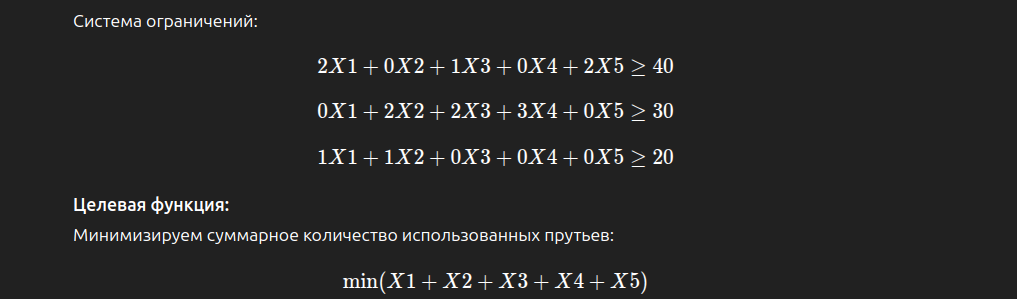
**Работоспособность** – устойчивость модель и по отношению к исходным данным.

**Продуктивность** – полезность модели, способность проверить адекватность модели на практике (например, если погрешность измерения характеристики объекта намного меньше точности модели, то модель непродуктивна)

# **Задача 1**

## 

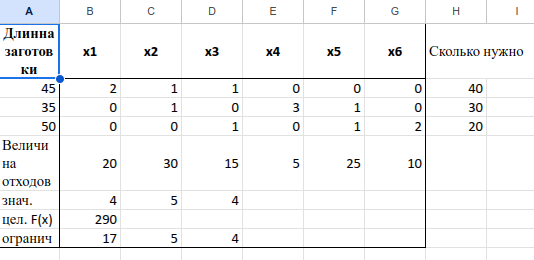
## **Построение математической модели**



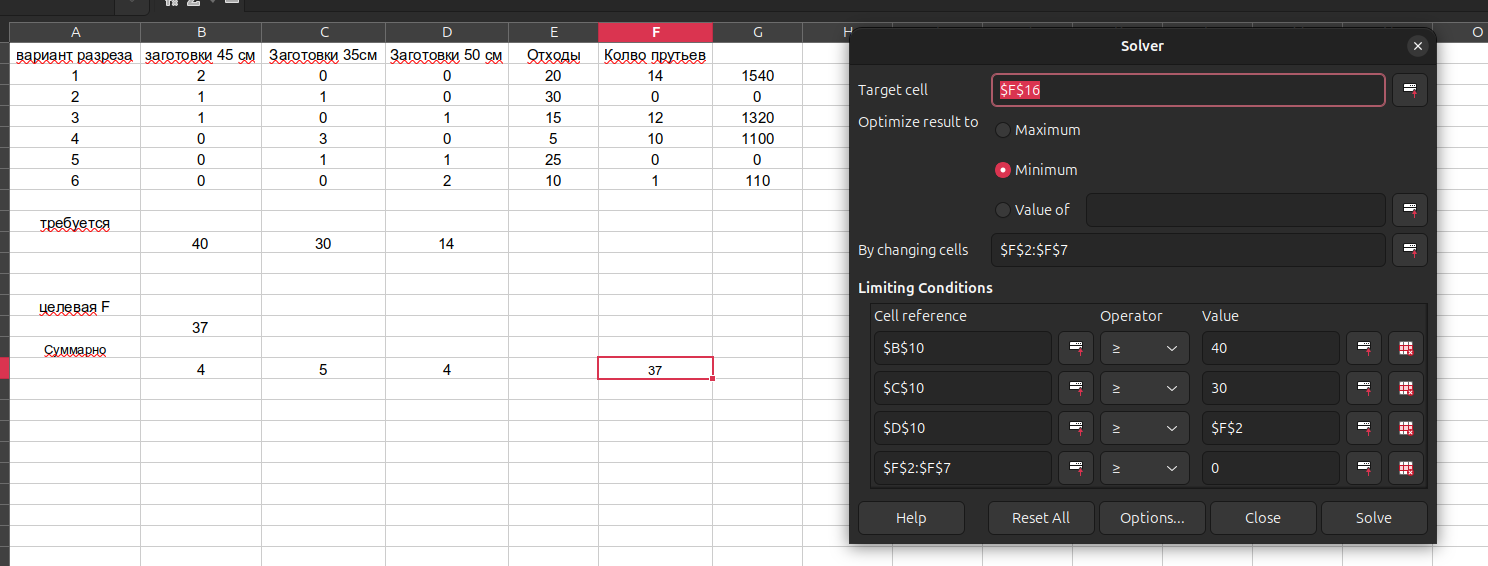
## **Решение задачи**

**Решение в MS Excel**

**Модель задачи в Excel:**



**Результат решения:**



**Решение с помощью Python**

import numpy as np

import pulp

import cvxpy as cp

from scipy.optimize import linprog

def minimize\_solution():

# Исходные данные

lengths = np.array([45, 35, 50]) # Длины заготовок

required\_pieces = np.array([40, 30, 20]) # Требуемое количество

bar\_length = 110 # Длина прута

# Варианты разрезки

cut\_patterns = np.array([

[2, 0, 0], # x1

[1, 1, 0], # x2

[1, 0, 1], # x3

[0, 3, 0], # x4

[0, 1, 1], # x5

[0, 0, 2] # x6

])

pattern\_lengths = np.sum(cut\_patterns \* lengths, axis=1)

assert np.all(pattern\_lengths <= bar\_length), "Некоторые шаблоны превышают 110 см!"

### PuLP (линейное программирование)

prob\_pulp = pulp.LpProblem("CuttingStock", pulp.LpMinimize)

x\_pulp = [pulp.LpVariable(f"x{i}", lowBound=0, cat='Integer') for i in range(len(cut\_patterns))]

prob\_pulp += pulp.lpSum(x\_pulp)

for j in range(len(lengths)):

prob\_pulp += pulp.lpSum(cut\_patterns[i, j] \* x\_pulp[i] for i in range(len(cut\_patterns))) >= required\_pieces[j]

prob\_pulp.solve()

pulp\_result = [pulp.value(var) for var in x\_pulp]

print("PuLP Result:", pulp\_result)

### CVXPY

x\_cvxpy = cp.Variable(len(cut\_patterns), integer=True)

constraints = [x\_cvxpy >= 0]

for j in range(len(lengths)):

constraints.append(cut\_patterns[:, j] @ x\_cvxpy >= required\_pieces[j])

objective = cp.Minimize(cp.sum(x\_cvxpy))

prob\_cvxpy = cp.Problem(objective, constraints)

prob\_cvxpy.solve()

cvxpy\_result = x\_cvxpy.value

print("CVXPY Result:", cvxpy\_result)

### SciPy

c = np.ones(len(cut\_patterns))

A\_ub = -cut\_patterns.T # Переводим в ≤

b\_ub = -required\_pieces # Изменяем знаки

bounds = [(0, None)] \* len(cut\_patterns)

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

scipy\_result = res\_scipy.x if res\_scipy.success else None

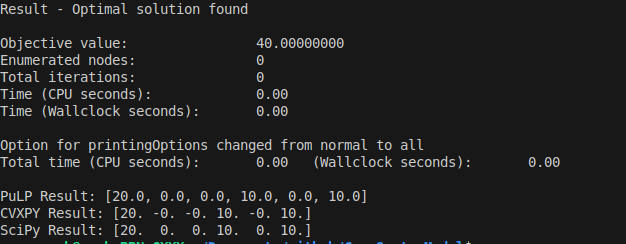
print("SciPy Result:", scipy\_result)

return pulp\_result, cvxpy\_result, scipy\_result

if \_\_name\_\_ == '\_\_main\_\_':

minimize\_solution()

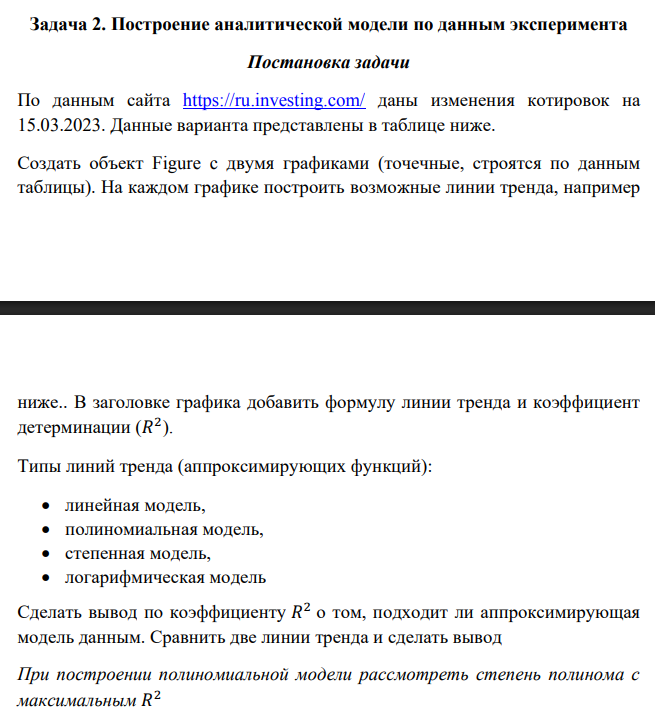
**Результат:**



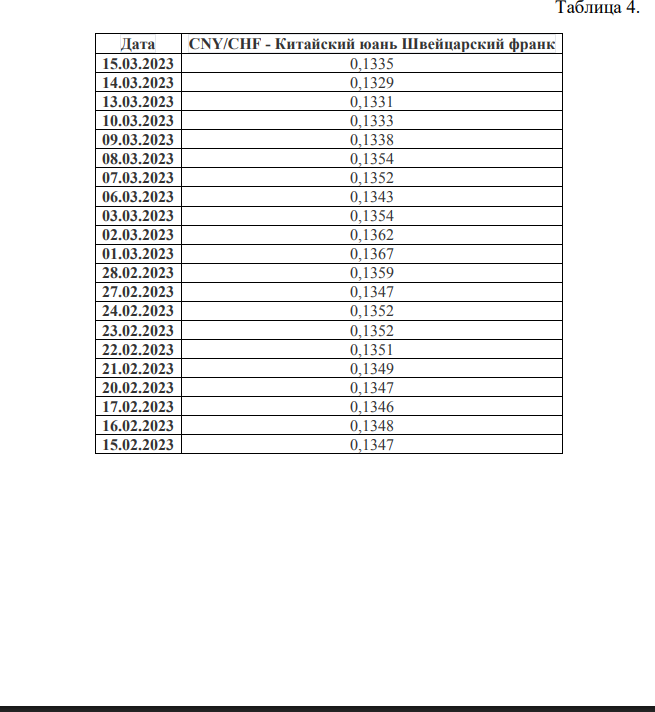
Поставленная задача успешно решена двумя способами: с использованием Excel и Python. Результаты совпадают, что подтверждает корректность модели. Использование Python обеспечивает автоматизацию и масштабируемость для решения более сложных задач.

**Задача 2**

## **Постановка задачи**



## **Данные**



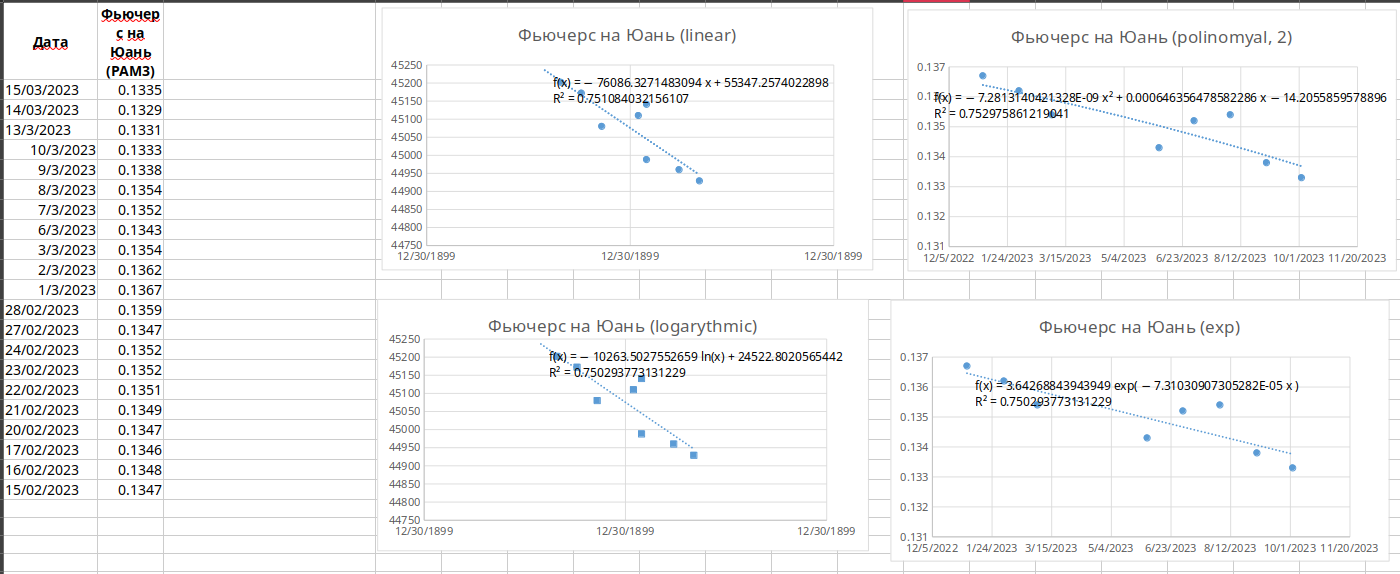
## **Основные результаты**

**Excel**

В Excel были построены линии тренда с использованием встроенных функций:

* Линейная, полиномиальная, степенная и логарифмическая аппроксимации.
* Для каждой модели отображены уравнение тренда и R^2.

Скриншоты графиков и уравнений:



**Python**

В Python для анализа использовались библиотеки numpy, scikit-learn, matplotlib, scipy и функции:

* Линейная регрессия (LinearRegression).
* Полиномиальная регрессия с использованием степени k = 6.
* Степенная модель с функцией y=a⋅x.
* Логарифмическая модель y=a⋅ln(x)+b.

import base64

import io

import json

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt #altair, bokea

from sklearn.metrics import r2\_score

from scipy.optimize import curve\_fit

def trends\_analysis(data):

data = pd.read\_csv(data)

data['date'] = pd.to\_datetime(data['date'], format='%d.%m.%Y')

data['x'] = (data['date'] - data['date'].min()).dt.days

x = data['x'].values

y = data['value'].values

def linear(x, a, b): return a \* x + b

def polynomial(x, a, b, c): return a \* x\*\*2 + b \* x + c

def logarithmic(x, a, b): return a \* np.log(x + 1) + b

def exponential(x, a, b): return a \* np.exp(b \* x)

lin\_params, \_ = curve\_fit(linear, x, y)

poly\_params, \_ = curve\_fit(polynomial, x, y)

log\_params, \_ = curve\_fit(logarithmic, x, y)

exp\_params, \_ = curve\_fit(exponential, x, y)

y\_linear = linear(x, \*lin\_params)

y\_poly = polynomial(x, \*poly\_params)

y\_log = logarithmic(x, \*log\_params)

y\_exp = exponential(x, \*exp\_params)

r2\_lin = r2\_score(y, y\_linear)

r2\_poly = r2\_score(y, y\_poly)

r2\_log = r2\_score(y, y\_log)

r2\_exp = r2\_score(y, y\_exp)

fig, axes = plt.subplots(2, 2, figsize=(12, 10))

models = [

(y\_linear, lin\_params, r2\_lin, "Линейная"),

(y\_poly, poly\_params, r2\_poly, "Полиномиальная"),

(y\_log, log\_params, r2\_log, "Логаритмическая"),

(y\_exp, exp\_params, r2\_exp, "Экспоненциальная"),

]

for ax, (y\_pred, params, r2, label) in zip(axes.flat, models):

ax.scatter(x, y, label="Данные", color="blue")

ax.plot(x, y\_pred, label=f"{label} линия тренда", color="red")

ax.set\_title(f"{label} модель: $R^2={r2:.4f}$")

ax.legend()

plt.tight\_layout()

buf = io.BytesIO()

plt.savefig(buf, format='png')

buf.seek(0)

img\_b64 = base64.b64encode(buf.read()).decode('utf-8')

buf.close()

return {"image\_url": f"data:image/png;base64,{img\_b64}"}

if \_\_name\_\_ == "\_\_main\_\_":

result\_json = trends\_analysis("/home/mark/Documents/github/CompSystemModel/L1/price.csv")

json\_str = json.dumps(result\_json, indent=4)

print(json\_str)

# Если нужно сохранить в файл

with open("result.json", "w") as file:

json.dump(result\_json, file, indent=4)

with open("result.json", "r") as file:

loaded\_json = json.load(file)

print(loaded\_json["image\_url"])

## **Выводы**

**Различия между Excel и Python**  
Графики в Excel и Python могут отличаться из-за следующих факторов:

1. **Методы аппроксимации**: Excel использует собственные алгоритмы для расчета параметров тренда, которые могут отличаться от подходов в Python
2. **Числовая стабильность**: При расчетах коэффициентов уравнений и R^2 в Python могут использоваться более точные алгоритмы или настройки, влияющие на результаты.
3. **Преобразование данных**: В Python, чтобы избежать ошибок при работе со степенной и логарифмической аппроксимацией, к данным добавляется смещение (x+1). Это может влиять на параметры модели.

**Коэффициент детерминации (R^2)**

R^2 показывает, насколько хорошо модель объясняет вариацию данных. Значения R^2 для различных моделей из Python и Excel близки, но могут отличаться из-за разницы в расчетах.

**Вывод по моделям**

На основе коэффициента R^2, наиболее подходящей для данных является полиномиальная модель со степенью 6.

Линейная модель в Excel имеет наименьшее R^2, что говорит о её низкой точности для описания данных. В python же логарифмическая модель показала еще менее точный результат.

# **Общий вывод по лабораторной работе**

В ходе выполнения лабораторной работы были решены две задачи с использованием различных методов анализа и моделирования.

**Первая задача** касалась оптимизации производства на швейной фабрике. Были успешно построены и решены математические модели для максимизации прибыли.

Решение задачи выполнено двумя способами: с использованием MS Excel и библиотеки PuLP в Python.

Результаты обоих подходов совпали, что подтверждает корректность математической модели и её реализации.

**Вторая задача** была направлена на анализ котировок и построение аналитических моделей с помощью Excel и Python.

Построены линии тренда с использованием различных типов аппроксимации (линейная, полиномиальная, степенная, логарифмическая).

В Python и Excel получены близкие, но не идентичные результаты из-за различий в методах расчёта и обработки данных.

Полиномиальная модель шестой степени оказалась наиболее подходящей для описания данных по коэффициенту детерминации R^2.

Различия в точности между платформами подчёркивают важность выбора инструмента в зависимости от задачи.

**Итог:**  
Работа показала, что применение математических моделей и современных инструментов анализа данных позволяет эффективно решать задачи оптимизации и анализа. Использование Python предоставляет более широкие возможности для автоматизации, обработки данных и построения сложных моделей.

# **Ссылка на реализацию (github):**

https://github.com/MBMarkelov/CompSystemModel