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»

|  |
| --- |
| Выполнил:  студент группы  Б.ПИН.РИС-21.06  Леонов А.М. |
| Проверила:  старший преподаватель кафедры ПО  Корнеева Е.И. |

Тверь 2024

**Оглавление**

**[Задание](#_Toc158808823)** [3](#_Toc158808823)

[**Теоретический вопрос:** 3](#_Toc158808824)

[**Задание 1. Построение аналитической модели по вербальному (текстовому) описанию.** 3](#_Toc158808825)

[**Задание 2. Построение аналитической модели по данным эксперимента.** 4](#_Toc158808826)

[**Решение** 6](#_Toc158808827)

[**Ответ на теоретический вопрос:** 6](#_Toc158808828)

[**Скриншоты работы программ:** 6](#_Toc158808829)

[**Ссылка на репозиторий и код программ** 7](#_Toc158808830)

[**Ссылка на репозиторий:** 7](#_Toc158808831)

[**Код:** 7](#_Toc158808832)

[**Краткий вывод по работе** 10](#_Toc158808833)

# **Задание**

## **Теоретический вопрос:**

Какая модель называется имитационной?

## **Задание 1. Построение аналитической модели по вербальному (текстовому) описанию.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вещество | Количество единиц вещества, содержащегося в 1 кг сырья вида | | | |
| **1** | **2** | **3** | **4** |
| A  B  C | 1  2  1 | 1  -  2 | -  3  4 | 4  5  6 |
| Цена 1 кг сырья (руб.) | 5 | 6 | 7 | 8 |

Из четырех видов сырья необходимо составить смесь, в состав которой должно входить не менее 26 ед. химического вещества A, 30 ед. – вещества B и 24 ед. – вещества C. Количество единиц химического вещества, содержащегося в 1 кг сырья каждого вида, указано в таблице. В ней же приведена цена 1 кг сырья каждого вида.

Составить смесь, содержащую не менее необходимого количества данного вида и имеющую минимальную стоимость.

x1-кол-во 1 сырья

x2-кол-во 2 сырья

x3-кол-во 3 сырья

x4-кол-во 4 сырья

F(x)=F(x1,x2,x3,x4)=x1\*5+x2\*6+x3\*7+x4\*4 ->Min -Основная функция

Ограничения:

x1\*1+x2\*1+x4\*4>=26

x1\*2+x3\*3+x4\*5>=30

x1\*1+x2\*2+x3\*4+x4\*6>=24

## **Задание 2. Построение аналитической модели по данным эксперимента.**

1. По экспериментальным данным построить график
2. Нарисовать линию тренда и по линии определить по какому закону изменяются данные (линейный, степенной (полиномиальный), экспоненциальный, логарифмический).
3. Определяем какая из линий больше подходит по коэффициент детерминации R2 . Выписать в отчет все варианты. Вывод - это какая линия тренда больше подходит данным и графику. ( R2 )
4. Вывод о том, какая функция аппроксимирует заданные данные (в зависимости от линии тренда и R2 )

Решение на python:

1. Ввести Xi, Yi в программу
2. Нарисовать график по данным
3. Выбрать график линии тренда (4 шт)
4. Добавить в область mathplotlib линию тренда по изменяемой функции
5. Рассчитать коэффициент R^2 (sklearn) <https://progler.ru/blog/python-koefficient-determinacii-r2>

|  |  |
| --- | --- |
| **Дата** | **Фьючерс на платину- Апр. ‘23(PLJ3)** |
| 15.03.2023 | 961,1 |
| 14.03.2023 | 997,3 |
| 13.03.2023 | 1004,9 |
| 12.03.2023 | 966,65 |
| 10.03.2023 | 962,2 |
| 09.03.2023 | 949,3 |
| 08.03.2023 | 940,6 |
| 07.03.2023 | 936,3 |
| 06.03.2023 | 978,6 |
| 05.03.2023 | 977,7 |
| 03.03.2023 | 979,4 |
| 02.03.2023 | 963,2 |
| 01.03.2023 | 961,8 |
| 28.02.2023 | 955,5 |
| 27.02.2023 | 941,9 |
| 26.02.2023 | 911,25 |
| 24.02.2023 | 907,6 |
| 23.02.2023 | 945,2 |
| 22.02.2023 | 953 |
| 21.02.2023 | 945,55 |
| 20.02.2023 | 929 |
| 19.02.2023 | 918,55 |
| 17.02.2023 | 921,4 |
| 16.02.2023 | 931 |
| 15.02.2023 | 917,8 |

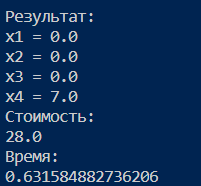
# **Решение**

## **Ответ на теоретический вопрос:**

Имитационная модель - это модель, которая воспроизводит поведение реальной системы или процесса. Такие модели используются для анализа и прогнозирования развития систем, а также для тестирования различных стратегий и решений. Имитационные модели могут быть как физическими (например, макет города для изучения движения транспорта), так и математическими (например, модель миграции населения).

## **Скриншоты работы программ:**

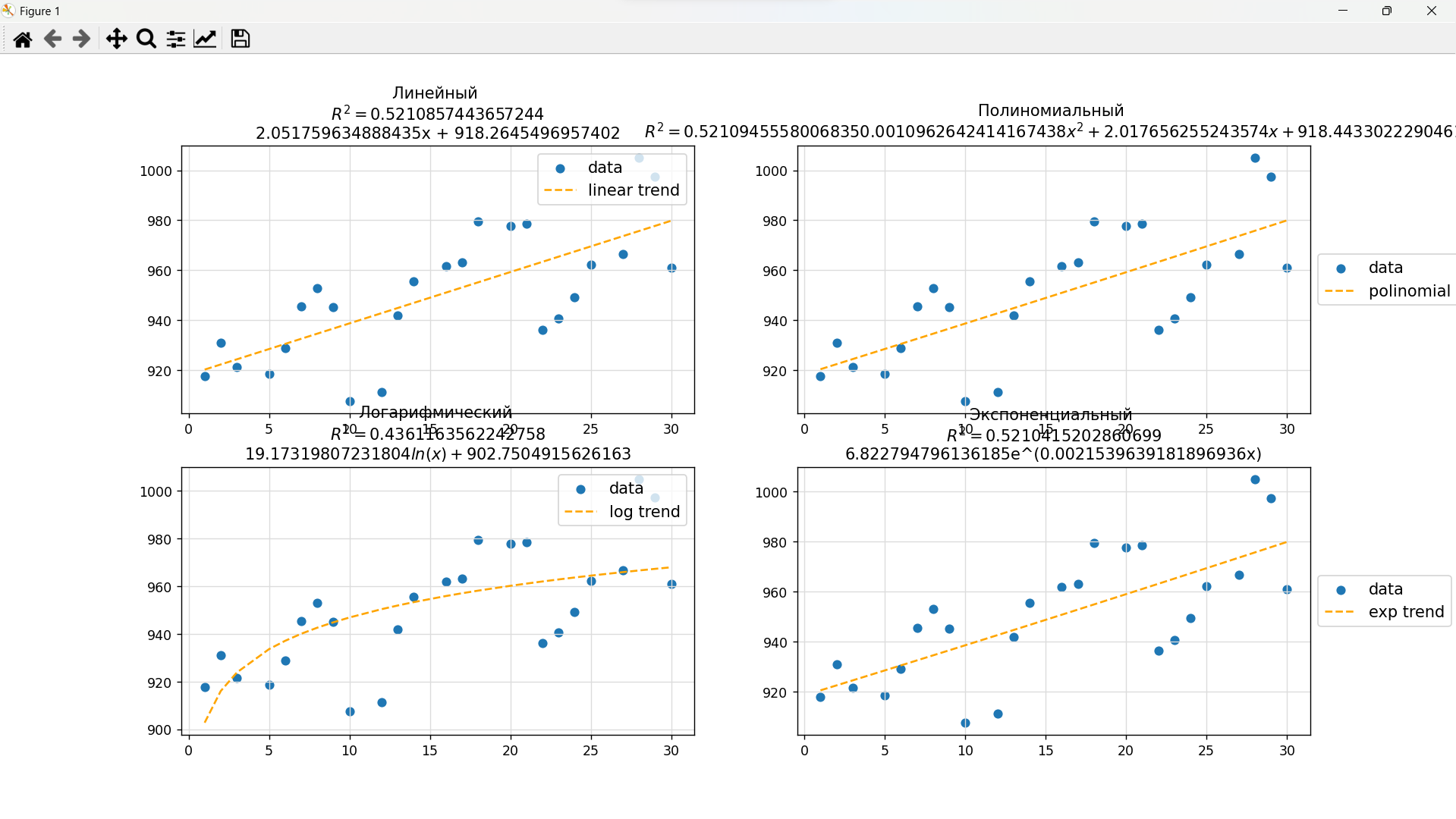
Задание 1, способ 1.



Задание 1, способ 2.



Задание 2.



# **Ссылка на репозиторий и код программ**

## **Ссылка на репозиторий:**

<https://github.com/akkafe1ix/ComputerSimulationLab1>

## **Код:**

Задание 1, способ 1.

import pulp

from pulp import LpProblem, LpMinimize, value

import time

start = time.time()

x1 = pulp.LpVariable("x1", lowBound=0)

x2 = pulp.LpVariable("x2", lowBound=0)

x3 = pulp.LpVariable("x3", lowBound=0)

x4 = pulp.LpVariable("x4", lowBound=0, cat='Integer')

# Создаем задачу линейного программирования для минимизации

problem = LpProblem("Minimize",LpMinimize)

# Определяем целевую функцию для минимизации

problem += x1\*5 + x2\*6 + x3\*7 + x4\*4

# Определяем ограничения

problem += x1 + x2 + 4 \* x4 >= 26

problem += 2\*x1 + 3\*x3 + 5\*x4 >= 30

problem += x1 + 2\*x2 + 4\*x3 + 6\*x4 >= 24

#problem += x4>=9

# Решаем задачу

problem.solve()

# Выводим результаты

print("Результат:")

for variable in problem.variables():

    print (variable.name, "=", variable.varValue)

print("Стоимость:")

print(value(problem.objective))

stop = time.time()

print ("Время:")

print(stop - start)

Задание 1, способ 2.

from scipy.optimize import linprog

# Коэффициенты целевой функции

c = [5, 6, 7, 4]

# Коэффициенты ограничений-неравенств (преобразованные в ≤, т.к. linprog минимизирует)

A = [[-1, -1, 0, -4],  # -x1 - x2 - 4x4 ≤ -26

     [-2, 0, -3, -5],  # -2x1 - 3x3 - 5x4 ≤ -30

     [-1, -2, -4, -6]] # -x1 - 2x2 - 4x3 - 6x4 ≤ -24

# Правые части ограничений-неравенств

b = [-26, -30, -24]

# Решение задачи линейного программирования

res = linprog(c, A\_ub=A, b\_ub=b, bounds=(0, None))

# Вывод результата

print("Оптимальное значение:", res.fun)

print("Оптимальные значения переменных:", res.x)

Задание 2.

# Полностью автоматическое создание трендов с matplotlib

import matplotlib.pyplot as plt # импортируем построитель графиков из библиотеки matplotlib

import numpy as np # импортируем библиотеку numpy для работы с массивами numpy

from sklearn.metrics import r2\_score # функция для расчёта критерия r^2

# график с точками

y = [917.8, 931, 921.4, 918.55, 929, 945.55, 953, 945.2, 907.6, 911.25, 941.9, 955.5, 961.8, 963.2, 979.4, 977.7, 978.6, 936.3, 940.6, 949.3, 962.2, 966.65, 1004.9, 997.30, 961.1]

x = [1,2,3,5,6,7,8,9,10,12,13,14,16,17,18,20,21,22,23,24,25,27,28,29,30]

# Массивы numpy по входным данным

numpy\_x = np.array(x)

numpy\_y = np.array(y)

# Линии тренда

# линейный (автоматическое создание)

set\_line\_by\_data = np.polyfit(numpy\_x, numpy\_y, 1) # полином первой степени

linear\_trend = np.poly1d(set\_line\_by\_data) # снижение размерности до одномерного массива

print("{0}x + {1}".format(\*set\_line\_by\_data)) # формула

# полиномиальный

# set\_polinom\_by\_data = np.polyfit(numpy\_x, numpy\_y, 6) # работа с полиномом 6 степени

# polinom\_trend = np.poly1d(set\_polinom\_by\_data) # Рассчитать значение полинома в точках x

# print("${0}x^6 + {1}x^5 + {2}x^4 + {3}x^3 + {4}x^2 + {5}x + {6}$".format(\*set\_polinom\_by\_data)) # формула

# set\_polinom\_by\_data = np.polyfit(numpy\_x, numpy\_y, 6)

set\_polinom\_by\_data = np.polyfit(numpy\_x, numpy\_y, 2) # работа с полиномом 2 степени

polinom\_trend = np.poly1d(set\_polinom\_by\_data) # Рассчитать значение полинома в точках x

polinom\_title = "${0}x^2 + {1}x + {2}$".format(\*set\_polinom\_by\_data)

print("${0}x^2 + {1}x + {2}$".format(\*set\_polinom\_by\_data)) # формула

# логарифмический

set\_log\_by\_data = np.polyfit(np.log(numpy\_x), numpy\_y, 1) # работа с полиномом 1 степени + логарифмирование x

log\_trend = [set\_log\_by\_data[0]\*np.log(x) + set\_log\_by\_data[1] for x in numpy\_x] # создание одномерного массива для логарифмического тренда

print("${0}ln(x) + {1}$".format(\*set\_log\_by\_data))  # формула

# экспоненциальный

set\_exp\_by\_data = np.polyfit(numpy\_x, np.log(numpy\_y), 1) # работа с полиномом 1 степени + логарифмирование

exp\_trend = [np.exp(set\_exp\_by\_data[1]) \* np.exp(set\_exp\_by\_data[0] \* x) for x in numpy\_x] # создание одномерного массива для экспоненциального тренда

print("${1}e^{0}x$".format(\*set\_exp\_by\_data))  # формула

# Расчёт R^2

linear\_r2 = r2\_score(numpy\_y, linear\_trend(numpy\_x))

polinom\_r2 = r2\_score(numpy\_y, polinom\_trend(numpy\_x))

log\_r2 = r2\_score(numpy\_y, log\_trend)

exp\_r2 = r2\_score(numpy\_y, exp\_trend)

# Отображение графиков

plt.figure(figsize=(15, 15)) # размер графика

# 2 графика по горизонтали, 2 по вертикали

plt.subplot(2, 2, 1)

# !!! Текущая ячейка - 1 (левый верхний график)

plt.scatter(numpy\_x, numpy\_y, label = 'data') # точечный график по x\_numpy, y\_numpy

plt.plot(numpy\_x, linear\_trend(numpy\_x), linestyle='dashed', color="orange", label = 'linear trend') # линейный тренд

plt.grid(color="gainsboro") # Сетка

plt.legend(loc='upper right', fontsize=12)

plt.title("Линейный \n$R^2=$" + str(linear\_r2) + "\n{0}x + {1}".format(\*set\_line\_by\_data))

# !!! Текущая ячейка - 2

plt.subplot(2, 2, 2)

plt.scatter(numpy\_x, numpy\_y, label = 'data') # точечный график по x\_numpy, y\_numpy

x = np.linspace(numpy\_x.min(), numpy\_x.max()) # набор данных для x для большей гладкости графика (50 точек)

plt.plot(x, polinom\_trend(x), linestyle='dashed', color="orange", label = 'polinomial trend') # полиномиальный тренд

plt.grid(color="gainsboro") # Сетка

plt.legend(loc = 'center left', fontsize=12, bbox\_to\_anchor=(1, 0.5))

plt.title("Полиномиальный \n$R^2=$" + str(polinom\_r2) + polinom\_title)

# !!! Текущая ячейка - 3

plt.subplot(2, 2, 3)

plt.scatter(numpy\_x, numpy\_y, label = 'data') # точечный график по x\_numpy, y\_numpy

plt.plot(numpy\_x, log\_trend, linestyle='dashed', color="orange", label = 'log trend') # логарифмический тренд

plt.grid(color="gainsboro") # Сетка

plt.legend(loc = 'upper right', fontsize=12)

plt.title("Логарифмический \n$R^2=$" + str(log\_r2) + "\n${0}ln(x) + {1}$".format(\*set\_log\_by\_data))

# !!! Текущая ячейка - 4

plt.subplot(2, 2, 4)

plt.scatter(numpy\_x, numpy\_y, label = 'data') # точечный график по x\_numpy, y\_numpy

plt.plot(numpy\_x, exp\_trend, linestyle='dashed', color="orange", label = 'exp trend')

plt.grid(color="gainsboro") # Сетка

plt.legend(loc = 'center left', fontsize=12, bbox\_to\_anchor=(1, 0.5))

plt.title("Экспоненциальный \n$R^2=$" + str(exp\_r2) + "\n{1}e^({0}x)".format(\*set\_exp\_by\_data))

fig = plt.gcf() # Взять текущую фигуру

fig.set\_size\_inches(15, 15) # Задать размеры графика

# Покажем окно с нарисованным графиком

plt.show()

# **Краткий вывод по работе**

Обе программы успешно работают на предоставленных в условиях данных. Задание 1 было выполнено с использованием двух способов, дабы проверить, что полученные значения оказались корректными и верными. Задание выполнено.