МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА № 43

ОТЧЁТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| доц., канд. техн. наук | |  |  | | | |  | | С. И. Колесникова |
| должность, уч. степень, звание | |  | подпись, дата | | | |  | | инициалы, фамилия |
| ОТЧЁТ О ЛАБОРАТОРНОЙ РАБОТЕ №4 | | | | | | | | | | |
| Модели статистического моделирования и прогнозирования динамических систем по временному ряду (на основе МНК) | | | | | | | | | | |
| по дисциплине: КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ | | | | | | | | | | |
|  | | | | | | | | | | |
| РАБОТУ ВЫПОЛНИЛА | | | | | | | | | | |
| СТУДЕНТКА ГР. | 4931 | | |  | 23.10.2022 |  | | Е.Ю. Ильченко | | |
|  |  | | |  | подпись, дата |  | | инициалы, фамилия | | |
|  |  | | |  |  |  | |  | | |

Санкт-Петербург 2022

1. **Цель работы:** Цель настоящей работы – освоить моделирование стохастических временных рядов.
2. Задание
3. Ознакомиться со справочными сведениями.
4. Сформулировать задачу МНК при построении функции регрессии.
5. Разработать программу, моделирующую алгоритм поиска оптимального решения для формализованной задачи, используя математический пакет MatLab или язык программирования Python:
   1. Самостоятельно реализовать МНК для решения задачи поиска коэффициентов модели, заданной в виде полинома второго порядка .
   2. С использованием встроенной реализации МНК в MatLab илиPython подобрать степень полиномиальной модели, наилучшим образом соответствующей исходным данным при визуальной оценке на графике. Для этого построить график с исходными данными (крестики, точки и т.п.) и различными вариантами полиномиальных моделей степени , где .
   3. Аппроксимировать данные функциональной моделью вида .
   4. Используя скорректированный коэффициент детерминации определить наилучшую из трех моделей , , .
6. Составить и представить преподавателю отчет о работе.

**Исходные данные:**

**Вариант 11**

Изучается динамика производства чугуна в стране. Для этого собраны данные об объемах его производства (млн.т.)*Y*(*t*) за 7 лет (1991-1996). Обосновать и построить тренд данного ряда. Оценить достоверность уточненной по МНК модели.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *t* | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| *Y*(*t*) | 107 | 108 | 107 | 110 | 111 | 110 | 112 |

1. **Ход выполнения задания**
2. Согласно методу наименьших квадратов (МНК) задача заключается в нахождении коэффициентов линейной зависимости, при которых функция трех переменных *а0*, *a1* и *a2* (принимает наименьшее значение). Решение примера сводится к нахождению экстремума функции трех переменных.
3. Вывод формул для нахождения коэффициентов *а0*, *a1* и *a2*. Функция будет принимать минимальное значение, если частные производные обращаются в 0. Составляется и решается система из трех уравнений с тремя неизвестными. Находим частные производные функции по переменным *а0*, *a1* и *a2*, приравниваем эти производные к нулю.

Получаем

Разделим на 2 и раскроем скобки

Таким образом мы получили систему из трех уравнений и трех неизвестных. Неизвестными выступают *а0*, *a1* и *a2 .*

Решим систему методом Крамера

Подставим в систему уравнений

, B =

*= 4676\*140\*7+784\*28\*140+140\*784\*28-140\*140\*140-28\*28\*4676-7\*784\*784=16464*

*= 15485\*140\*7+784\*28\*765+140\*3083\*28-765\*140\*140-28\*28\*15485-7\*3083\*784=196*

*= 4676\*3083\*7+15485\*28\*140+140\*784\*765-140\*3083\*140-765\*28\*4676-7\*784\*15485=11956*

*= 4676\*140\*765+784\*3083\*140+15485\*784\*28-140\*140\*15485-28\*3083\*4676-765\*784\*784=1747536*

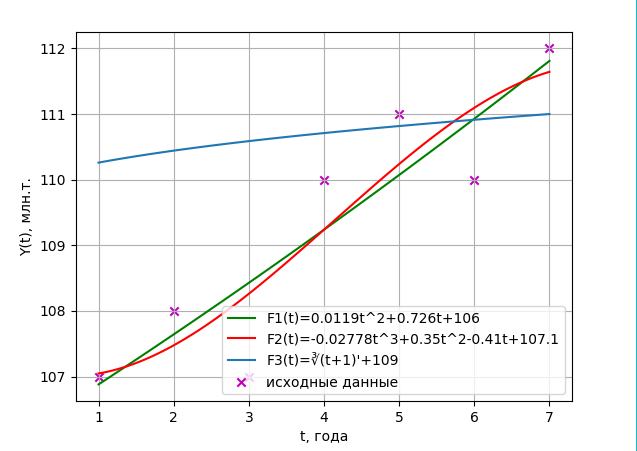
Данные полученные после выполнение работы программы:

1. Выражение для модели, найденное с помощью самостоятельно реализованного метода МНК и средствами математического пакета Python

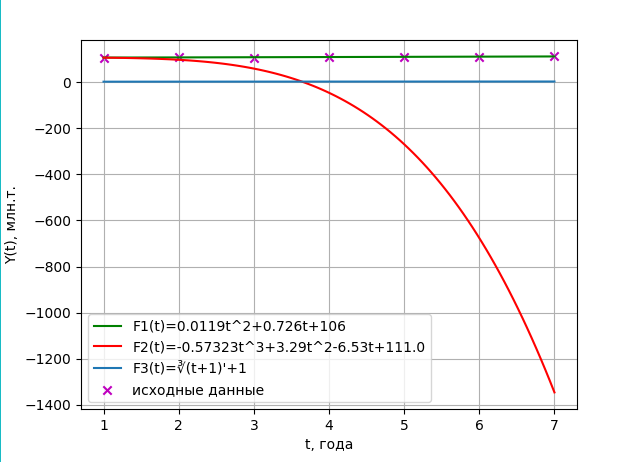
Модели получены в результате письменных вычислений и программирования совпали. Модель МНК, найденная с помощью самостоятельной реализации метода МНК имеет вид:

1. Выражение для модели, найденное с использованием встроенной реализации МНК в Python путём подбора степени полиномиальной модели, наилучшим образом соответствующей исходным данным при визуальной оценки на графике. Для этого были построены графики с исходными данными и различные варианты полиномиальных моделей степени p, где p≠2 (Была использована функция polyfit)

p=3



p=4



Наилучший результат получен при степени полинома 3

1. Для ответа на вопрос: какая из кривых лучше аппроксимирует исходные данные, следует оценить погрешности аппроксимаций.

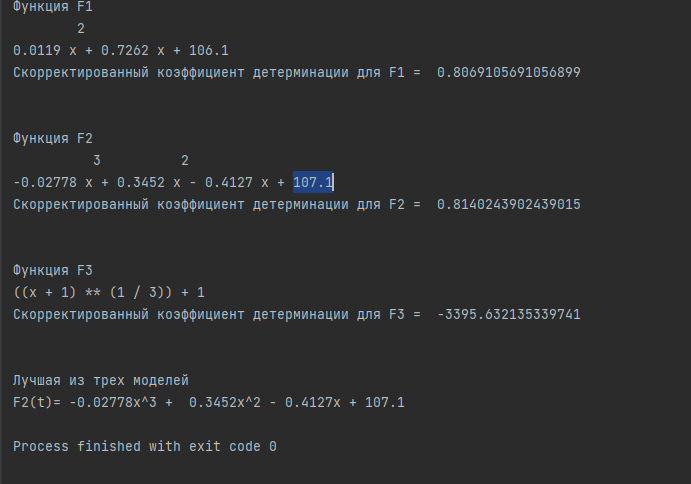
Коэффициент детерминации:

где – условная дисперсия ошибки модели, исправленная нормирующим коэффициентом, – дисперсия случайной величины . Здесь – результат *j*-ой модели в точке , – количество наблюдений за переменными и , – количество параметров *j*-ой модели.

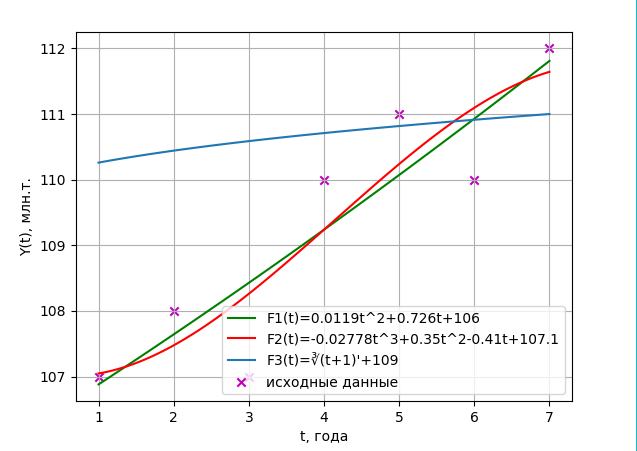
Cкорректированный коэффициент детерминации, лишенный данного недостатка:

Результаты вычисления скорректированного коэффициента детерминации для трех моделей:

**Вывод**: кривая является лучшим приближением исходных данных по сравнению с другими кривыми.



Это также видно из графика



**Выводы**

В ходе выполнения лабораторной работы были освоены средства моделирования стохастических временных рядов, аппроксимированы данные функциональными моделями трех видов, определена наилучшая модель – полином второго порядка:

**Листинг программы**

import matplotlib.pyplot as plt  
import numpy as np  
  
# Данные по варианту (рождаемость в России)  
t = np.array([1, 2, 3, 4, 5, 6, 7], dtype='int64')  
y = np.array([107, 108, 107, 110, 111, 110, 112])  
  
  
# Самостоятельно реализованный МНК  
def fun\_MNK(y, t):  
 # Находим ∑(ti ^ 4)  
 sum1 = 0  
 for i in range(0, len(t)):  
 sum1 += (t[i] \*\* 4)  
  
 # Находим ∑(ti ^ 3)  
 sum2 = 0  
 for i in range(0, len(t)):  
 sum2 += (t[i] \*\* 3)  
  
 # Находим ∑(ti ^ 2)  
 sum3 = 0  
 for i in range(0, len(t)):  
 sum3 += (t[i] \*\* 2)  
  
 # Находим ∑(ti)  
 sum4 = sum(t)  
  
 # Находим ∑(ti ^ 2 \* y)  
 sum5 = 0  
 for i in range(0, len(t)):  
 sum5 += (t[i] \*\* 2 \* y[i])  
  
 # Находим ∑(ti \* y)  
 sum6 = 0  
 for i in range(0, len(t)):  
 sum6 += (t[i] \* y[i])  
  
 # Находим ∑(y)  
 sum7 = sum(y)  
  
 # Матрица  
 A = np.array([[sum1, sum2, sum3],[sum2, sum3, sum4],[sum3, sum4, 7]])  
 # Столбец свободных членов  
 B = np.array([sum5, sum6, sum7]).reshape((3, 1))  
  
 # Решаем систему матричным методом  
 res = np.linalg.solve(A, B)  
 f1 = np.poly1d([res[0][0], res[1][0], res[2][0]])  
 return f1  
  
# Поиск скорректированного коэффициента детерминации  
def R2\_adj(y\_t, y\_p):  
 # Количество наблюдений за переменными t и y  
 n = len(y\_t)  
 # Количество параметров j-ой модели  
 k = 1  
 # Условная дисперсия ошибки модели, исправленная нормирующим коэффициентом  
 Dyx=sum(map(lambda y\_teor,y\_pract:(y\_teor-y\_pract)\*\*2, y\_t, y\_p))  
 y\_avarage= sum(y\_t) / len(y\_t)  
 # Дисперсия СВ y  
 Dy=sum(map(lambda y\_teor:(y\_teor-y\_avarage)\*\*2, y\_t))  
 # Коэффициент детерминации  
 R=1-(Dyx/Dy)  
 # Скорректированный коэффициент детерминации  
 R2adj=1-(1-R)\*(n-1)/(n-k)  
 return R2adj  
  
if \_\_name\_\_ == '\_\_main\_\_':  
  
 # Функция F1(t)  
 F1 = fun\_MNK(y, t)  
 R1=R2\_adj(y, F1(t))  
 print("Функция F1")  
 print(F1)  
 print("Скорректированный коэффициент детерминации для F1 = ", R1)  
 print("\n")  
  
 #Функция F2(t), реализованная при помощи numpy.polyfit (4 степень полинома)  
 c = np.polyfit(t, y, 3)  
 F2 = np.poly1d(c)  
 R2 = R2\_adj(y, F2(t))  
 print("Функция F2")  
 print(F2)  
 print("Скорректированный коэффициент детерминации для F2 = ", R2)  
 print("\n")  
  
 # Третья функция F3 (t)=∛(t+1)'+1  
 F3 = ((t + 1) \*\* (1 / 3)) + 1  
 R3 = R2\_adj(y, F3)  
 print("Функция F3")  
 print("((x + 1) \*\* (1 / 3)) + 1")  
 print("Скорректированный коэффициент детерминации для F3 = ", R3)  
 print("\n")  
  
 # Определение наилучшей из трех моделей F1(t), F2(t), F3(t)  
 # Наилучшая модель-модель, у которой R близок к 1  
 print("Лучшая из трех моделей")  
 if abs(1 - R2) >= abs(1 - R1) <= abs(1 - R3):  
 print("F1(t)= 0.0119x^2 + 0.7262x - 106.1")  
 elif abs(1 - R1) >= abs(1 - R2) <= abs(1 - R3):  
 print("F2(t)= -0.02778x^3 + 0.3452x^2 - 0.4127x + 107.1")  
 else:  
 print("F3(t)=∛(t+1)'+1")  
  
 # Вывод графиков  
 plt.ylabel('Y(t), млн.т.')  
 plt.xlabel('t, года')  
 x = np.linspace(1, 7, 100) # От 1 до 7  
  
 # График для функции F1  
 f1 = F1[2] \* (x \*\* 2) + F1[1] \* x + F1[0]  
 plt.figure(1)  
 plt.plot(x, f1, color='g', label="F1(t)={}t^2+{}t+{}".format(round(F1[2], 4), round(F1[1], 3), round(F1[0])))  
  
 # График для функции F2  
 f2 = F2[3] \* (x \*\* 3) + F2[2] \* (x \*\* 2) + F2[1] \* x + F2[0]  
 plt.figure(1)  
 plt.plot(x, f2, label="F2(t)={}t^3+{}t^2{}t+{}".format(round(F2[3], 5), round(F2[2], 2), round(F2[1], 2), round(F2[0], 1)), color='r', )  
  
 # График для функции F3  
 F3 = ((x + 1) \*\* (1 / 3)) + 1  
 #print(f3)  
 plt.figure(1)  
 plt.plot(x, F3, label="F3(t)=∛(t+1)'+1")  
  
 # Вывод исходных данных  
 plt.scatter(t, y, label="исходные данные", color='m', marker='x')  
  
 plt.legend()  
 plt.grid()  
 plt.show()