МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»**

(БГТУ им. В.Г.Шухова)

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Системный анализ и обработка информации

Лабораторная работа №5

**Регрессионный анализ**

Выполнил:

ст. гр. ВТ-31

Клесов М.И.

Проверил:

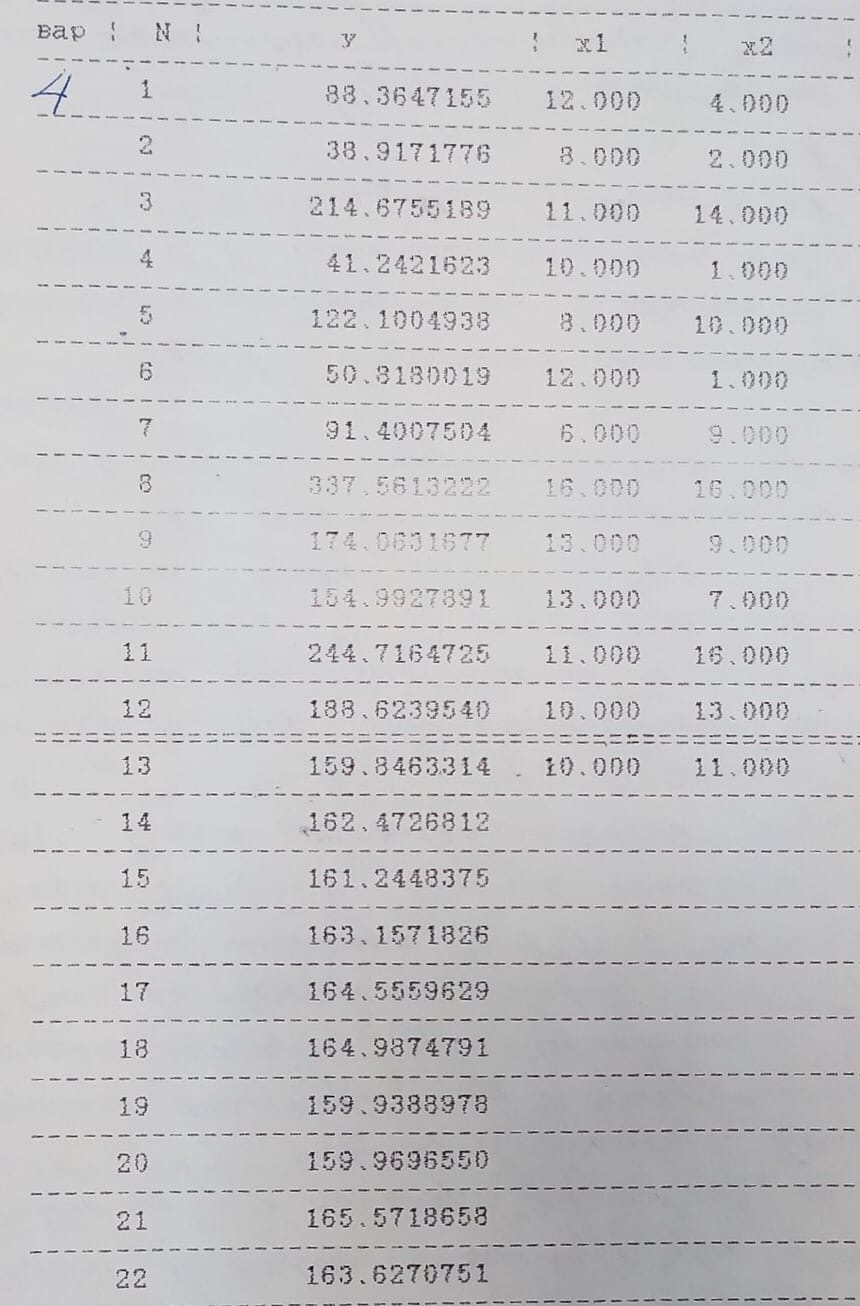
Полунин А.И.

Белгород, 2020

**Цель работы**: получить математическую модель процесса, заданного его числовыми значениями при разных значениях двух аргументов. Числовые значения процесса измеряются со случайной ошибкой, распределенной по нормальному закону, мат. ожидание ошибки равно нулю, дисперсия неизвестна. Процесс описывается алгебраической зависимостью.

**Задание**: в таблице находятся результаты эксперимента для определения функции, зависящей от двух аргументов. Эксперименты 1 – 12 использовать для оценки коэффициентов математической модели , эксперименты 13 – 22 использовать для оценки дисперсии погрешности измерений .

Задание варианта № 4



Алгоритм выполнения

Считаем, что аппроксимируемый процесс имеет вид

, где — оценки неизвестных параметров, — регрессоры.

На основании экспериментальных данных составляют матрицу:

F = - матрица регрессоров, где — значение j-го регрессора в i-ом эксперименте.

F =

[[ 12. 4. 48.]

[ 8. 2. 16.]

[ 11. 14. 154.]

[ 10. 1. 10.]

[ 8. 10. 80.]

[ 12. 1. 12.]

[ 6. 9. 54.]

[ 16. 16. 256.]

[ 13. 9. 117.]

[ 13. 7. 91.]

[ 11. 16. 176.]

[ 10. 13. 130.]]

Вектор оцениваем по методу наименьших квадратов

где Y – вектор оценки значений исследуемого процесса, полученный в экспериментах.

Для этого вычислим среднее арифметическое результатов эксперимента:

и величины , где — значение модели, полученное при наборе регрессоров, вычисленных для i-го эксперимента.

A =

[[2.95952526]

[1.94973939]

[1.0191427 ]]

Математическое ожидание теоретически = 145.623043825

Проведём оценку качества математической модели, для этого вычислим значение критерия Фишера на уровне значимости 0.95:

, значит коэффициент множественной корреляции R значим, математическая модель процесса содержит аргументы действительного процесса и необходимо дальше уточнять модель.

Проверим значимость коэффициентов уравнения регрессии. Для этого оценим по результатам экспериментов дисперсию неизвестной погрешности ε (дисперсию ошибок измерений), влияющей на эксперимент:

Вычислим дисперсии оценки коэффициентов математической модели. Для этого используем зависимости

=(FTF)-1, = .

Для проверки значимости вычисленного коэффициента вычисляем величину



Коэффициент Стьюдента: F(0.95,9)= tα =2.2621

Все ****, значит коэффициент  нельзя объяснить действием случайных факторов, он значим и его можно использовать в модели.

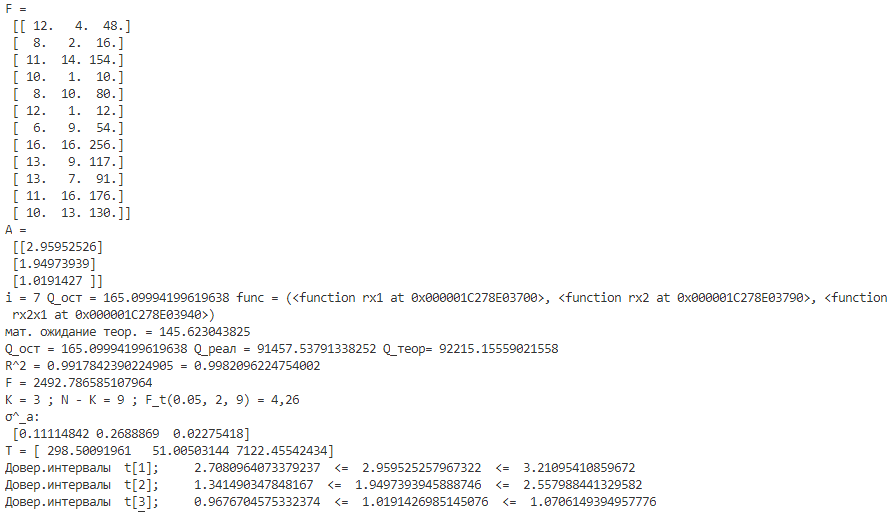
Вычислим доверительные интервалы для коэффициентов уравнения регрессии.

Довер.интервалы t[1]; 2.7080964073379237 <= 2.959525257967322 <= 3.21095410859672

Довер.интервалы t[2]; 1.341490347848167 <= 1.9497393945888746 <= 2.557988441329582

Довер.интервалы t[3]; 0.9676704575332374 <= 1.0191426985145076 <= 1.0706149394957776

Результат работы программы



Текст программы

from itertools import \*

import numpy as np

import matplotlib.pyplot as plt

import math

import scipy

import scipy.optimize as opt

import scipy.integrate as integrate

def r0(x1,x2):

  return 0

def r1(x1,x2):

  return 1

def rx1(x1,x2):

  return x1

def rx2(x1,x2):

  return x2

def rx1\_2(x1,x2):

  return x1\*x1

def rx2\_2(x1,x2):

  return x2\*x2

def rx2x1(x1,x2):

  return x2\*x1

arm=[rx1,rx2,rx2\_2,rx2x1]

def Get\_F(i:int,RRR):#возвр.матр. F

  count\_r=len(RRR[i])

  F=np.zeros((n1,count\_r  ))#M: 12 x кол-во регрессоров

  for it in range(n1):

    for j in range(count\_r):

      F[it][j]=RRR[i][j](X1[it][0],X2[it][0])

  return F

def Get\_res\_s(it:int,A):

  xx0=[]

  xx1=[]

  xx2=[]

  yy0=[]

  yynew=[]

  for i in range(n1):

    xx0.append(i)

    xx1.append(X1[i][0])

    xx2.append(X2[i][0])

    yy0.append(Y0[i][0])

    tmp=0.0

    for j in range(len(A)):

      tmp+=A[j][0]\*RRR[it][j](X1[i][0],X2[i][0])  #Y^=Fa^

    yynew.append(tmp) #y^=sum<i=1, k>(a^\_i\*f\_i(x))

  return xx0,xx1,xx2,yy0,yynew

def Get\_Grafic(xx0,yy0,yynew):

  fig, ax = plt.subplots()

  ax.scatter(xx0, yy0)

  ax.plot(xx0, yy0, 'r', lw=2, label="Theoretical")

  ax.plot(xx0, yynew, 'b', lw=2, label="Fit")

  ax.legend()

  ax.set\_xlim(0, 13)

  ax.set\_xlabel(r"$x$", fontsize=18)

  ax.set\_ylabel(r"$y$", fontsize=18)

  plt.show()

  return

n1=12

n2=10

##вектор x1,x2,y

X1=np.array([

   [12.0],

   [8.0],

   [11.0],

   [10.0],

   [8.0],

   [12.0],

   [6.0],

   [16.0],

   [13.0],

   [13.0],

   [11.0],

   [10.0],

   [10.0]

])

X2=np.array([

   [4.0],

   [2.0],

   [14.0],

   [1.0],

   [10.0],

   [1.0],

   [9.0],

   [16.0],

   [9.0],

   [7.0],

   [16.0],

   [13.0],

   [11.0]

])

Y0=np.array([

   [88.3647155],

   [38.9171776],

   [214.6755189],

   [41.2421623],

   [122.1004938],

   [50.8180019],

   [91.4007504],

   [337.5613222],

   [174.0631677],

   [154.9927891],

   [244.7164725],

   [188.6239540],

])

Y=np.array([

   159.8463314,

   162.4726812,

   161.2448375,

   163.1571826,

   164.5559629,

   164.9874791,

   159.9388978,

   159.9696550,

   165.5718658,

   163.6270751

])

def Get\_Qo(yy0,yynew):

  Qo=0.0 #Q остаток

  for i in range(n1):

    Qo+=(yy0[i]-yynew[i])\*\*2

  return Qo

def Get\_RRR():

  tf=[]

  for i in range(2, len(arm)):

    j = combinations(arm, i)

    tf+=list(j)

  return tf

global r\_min

r\_min=999999.9

global r\_num

r\_num=0

RRR=Get\_RRR()

for iterat in range(len(RRR)):

  F=Get\_F(iterat,RRR)

  A=(np.linalg.inv(((F.transpose()).dot(F))).dot(F.transpose())).dot(Y0)

  xx0,xx1,xx2,yy0,yynew=Get\_res\_s(iterat,A)

  Qoo=Get\_Qo(yy0,yynew)

  print('i =', iterat,'Q\_ост =', Qoo)

  if(r\_min>Qoo):

    r\_min=Qoo

    r\_num=iterat

F=Get\_F(r\_num,RRR)

print('F =\n',F)

A=(np.linalg.inv(((F.transpose()).dot(F))).dot(F.transpose())).dot(Y0)  #a^

print('A =\n',A)

xx0,xx1,xx2,yy0,yynew=Get\_res\_s(r\_num,A)

print('i =', r\_num, 'Q\_ост =', r\_min, 'func =', RRR[r\_num])   #STR\_RRR

Get\_Grafic(xx0,yy0,yynew)

############################################################################

my\_t=np.mean(yy0)#MY теор

print('мат. ожидание теор. =',my\_t)

Q=0.0  #Q теор

Qr=0.0 #Q реал

Qo=0.0 #Q остаток

for i in range(n1):

  Q += (yy0[i]-my\_t)\*\*2

  Qr += (yynew[i]-my\_t)\*\*2

  Qo += (yy0[i]-yynew[i])\*\*2

R\_2=Qr/Q

R\_22=1.0-Qo/Q

print('Q\_ост =', Qo,'Q\_реал =', Qr,'Q\_теор=', Q)

print('R^2 =',R\_2,'=',R\_22)

K = len(A)  #кол-во регрессоров

Sr\_2 = Qr/(K-1)

So\_2 = Qo/(n1-K)

FF = Sr\_2/So\_2

print('F =', FF)

print('K =', K, '; N - K =', n1 - K, '; F\_t({}, {}, {}) = 4,26'.format(0.05, K-1, n1-K))

y\_13\_22=np.average(Y)

q\_e\_2=0.0 #(σ^)^2\_e

l=len(Y)

for i in range(l):

  q\_e\_2+=((Y[i]-y\_13\_22)\*\*2)/(l-1)

q\_e\_2=np.sqrt(q\_e\_2)

#print(F)

#print(q\_e\_2)

C = np.abs(np.linalg.inv(np.dot(F.transpose(),F)))

q\_a = np.zeros(len(C))

for i  in range(len(q\_a)):

  q\_a[i] = np.sqrt(C[i][i])\*q\_e\_2

print('σ^\_a:\n', q\_a)

q\_a = q\_a\*\*2

T = np.zeros(len(q\_a))

sk = np.abs(np.sqrt(A[0][0]\*A[0][0]+A[1][0]\*A[1][0]+A[2][0]\*A[2][0]))

for i in range(len(q\_a)):

  T[i] = sk/q\_a[i]

print('T =', T)

q\_a = np.sqrt(q\_a)

t\_r = 2.2621

for i in range(len(q\_a)):

  print("Довер.интервалы  t[{}];".format(i+1), "   ", A[i][0]-t\_r\*q\_a[i], " <= ", A[i][0], " <= ", A[i][0]+t\_r\*q\_a[i])