

Министерство науки и высшего образования Российской Федерации
Новосибирский государственный технический университет

Статистические методы анализа данных
Лабораторная работа №3

Факультет:	ФПМИ
Группа:	ПМ-63
Студенты:	Кожекин М.В. Майер В.А. Назарова Т.А. Утюганов Д.С.
Вариант:	1

Новосибирск

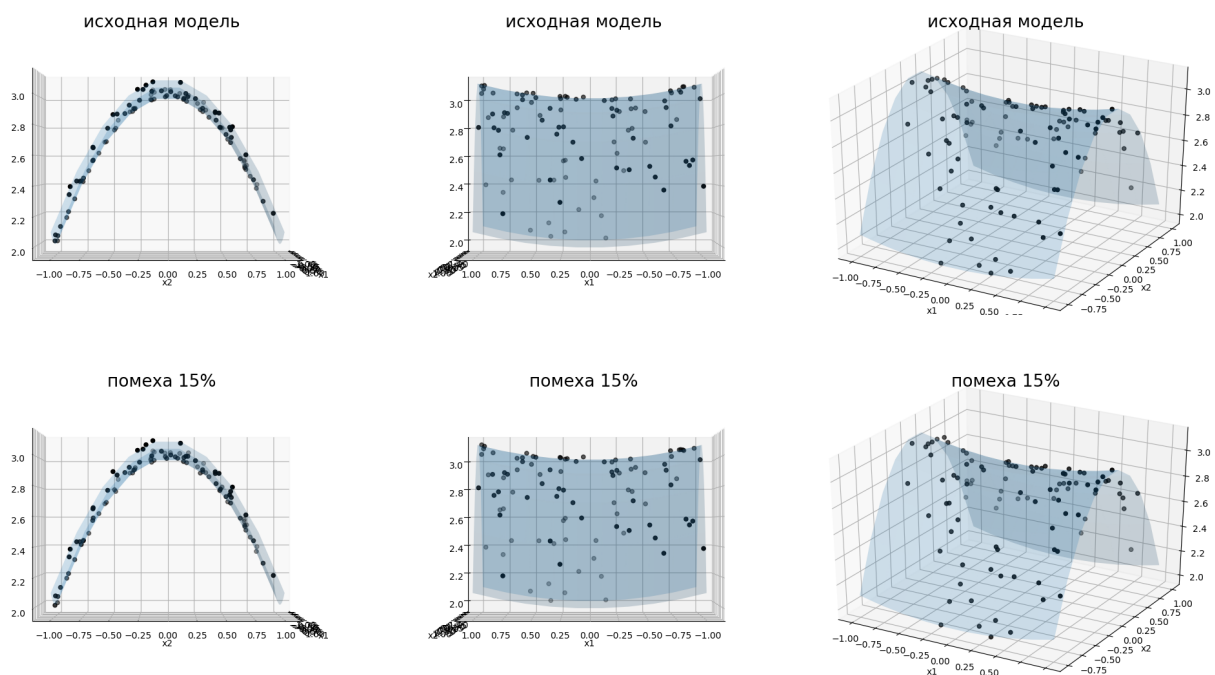
2019

1. Цель работы

Разработать программу для генерации экспериментальных данных по схеме имитационного моделирования и решить обратную задачу.

2. Решение

1. Изменить модель регрессии, добавив в неё дополнительный регрессор, ранее не входивший в состав модели, порождающей данные. Не генерируя новых данных, найти точечные оценки всех параметров расширенной модели. В дальнейшем при рассмотрении этой расширенной модели анализе должно быть показано, что параметр при дополнительном регрессоре незначим.



2,3. Построить доверительные интервалы для каждого параметра модели и проверить гипотезу о его значимости

Доверительный интервал для каждого параметра θ представим в виде двустороннего неравенства:

$$\hat{\theta}_j - t_{/2, f_R} \leq \theta_j \leq \hat{\theta}_j + t_{/2, f_R}$$

Для определения значимости параметра необходимо посчитать статистику $F = \frac{(\hat{\theta}_j)^2}{\sigma^2 d_{jj}}$ и её критическое значение $F_{\alpha, 1, n-m}$

$\hat{\theta}_j - t_{/2, f_R} \sigma(\hat{\sigma}_j)$	$\hat{\theta}_j$	$\hat{\theta}_j + t_{/2, f_R} \sigma(\hat{\sigma}_j)$	F	$F_{\alpha, q, n-m}$	значимость параметра
-0.117	0.103	0.323	741.620	3.940	+
-1.265	-1.005	-0.744	59,607.115	3.940	+
-0.061	-0.001	0.059	0.416	3.940	-
2.927	3.001	3.075	1,863,620.789	3.940	+

Как мы видим новый регрессор оказался незначим.

4. Проверить гипотезу о незначимости самой регрессии

$$F = \frac{(RSS_H - RSS)/q}{RSS/(n - m)}$$

$$RSS = (y - X\hat{\theta})^T (y - X\hat{\theta})$$

$$RSS_H = \sum_{i=1}^n (y_i - \bar{y})^2$$

$$q = m - 1$$

Гипотеза о незначимости регрессии отвергается, т.к. $F = 21093.487 > F_{\alpha, q, n-m} = 2.699$

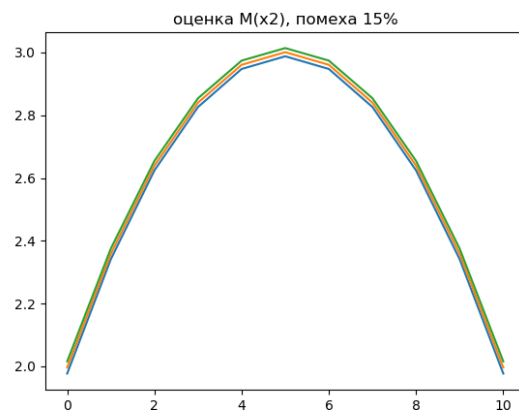
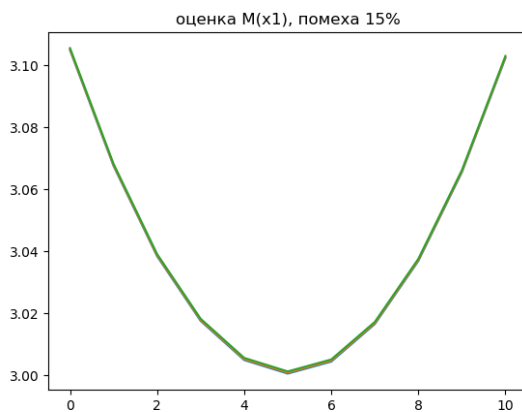
5. Рассчитать прогнозные значения для математического ожидания функции отклика $\eta(x, \hat{\theta}) = f^T(x)\hat{\theta}$ для всего интервала действия одного из факторов, зафиксировав значения других факторов на границе или в центре области их определения.

6. По полученным в п.5 прогнозным значениям построить графики прогнозных значений и доверительной полосы для математического ожидания функции отклика и для самого отклика.

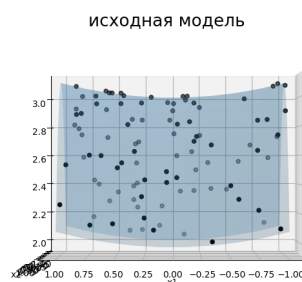
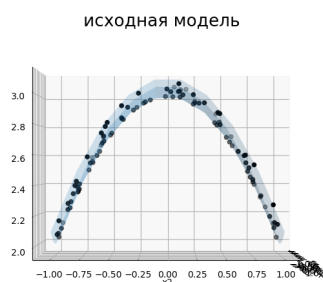
Для оценки математического ожидания функции отклика построим доверительный интервал:

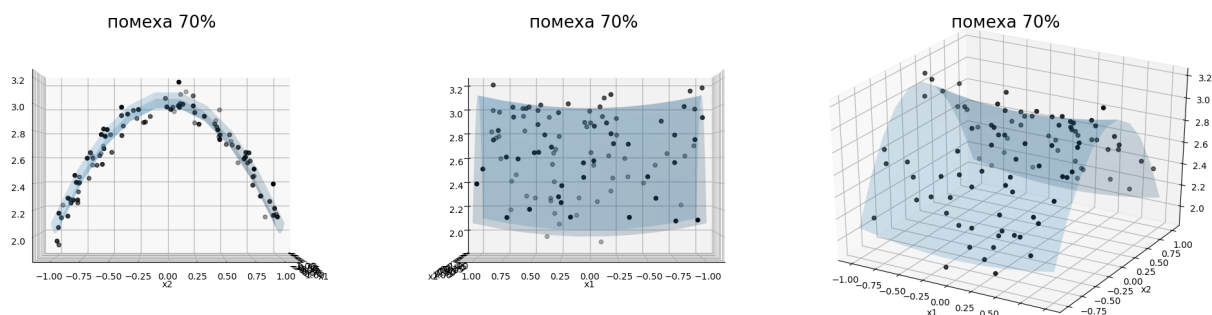
$$\eta(x, \hat{\theta}) - t_{/2, f_R} \sigma(\eta(x, \hat{\theta})) \leq \eta(x, \hat{\theta}) \leq \eta(x, \hat{\theta}) + t_{/2, f_R} \sigma(\eta(x, \hat{\theta}))$$

$$\sigma(\eta(x, \hat{\theta})) = \hat{\sigma} \sqrt{1 + f^T(x)(X^T X)^{-1} f(x)}$$



7. Заново смоделировать исходные данные (см. лаб. работу №1), увеличив мощность случайной помехи до 50...70% от мощности полезного сигнала, и провести оценку параметров. Повторить пункты 3, 4 с новыми данными.





$\hat{\theta}_j - t_{/2, f_R} \sigma(\hat{\sigma}_j)$	$\hat{\theta}_j$	$\hat{\theta}_j + t_{/2, f_R} \sigma(\hat{\sigma}_j)$	F	$F_{\alpha, q, n-m}$	значимость параметра
-0.111	0.111	0.334	23.717	3.940	+
-1.215	-0.993	-0.771	1,889.521	3.940	+
-0.071	-0.004	0.062	0.127	3.940	-
2.912	2.993	3.074	46,930.283	3.940	+

Как видно, даже при таком высоком уровне шума модель адекватно оценивает параметры.

3. Исходный код программы

Файл run_lab3.py

```

1 import os
2
3 os.system('python 1/lab1.py 0.15 3')
4 os.system('python 2/lab2.py 0.15 4')
5 os.system('python 3/lab3.py 0.15 4')
6
7 os.system('python 1/lab1.py 0.70 3')
8 os.system('python 2/lab2.py 0.70 4')
9 os.system('python 3/lab3.py 0.70 4')

```

Файл model_3parameters.py

```

1 # файл параметров модели
2
3 theta_true = [0.1, -1, 3]          # параметры уравнения  $t_0*x_1^2 + t_1*x_2^2 + t_2$ 
4 a, b, n, m = -1, 1, 100, 3        # границы, число точек и параметров
5 rho = 0.15                        # шум в диапазоне [0.05, 0.15] или [0.50, 0.70]
6 alpha = 0.05                      # уровень значимости
7 dist_E = 0.000491836210109302    # для  $\theta_{true} = [-0.05, 2, 4]$ 
8 f_E = 9000                         # вместо float('inf')
9 count = 11
10
11
12 def f_2_parameters(th, x1, x2):
13     return th[0]*x1**2 + th[1]*x2**2 + th[2]
14
15
16 def f(th, x_vector):
17     x1 = float(x_vector[0])
18     x2 = float(x_vector[1])
19     return th[0]*x1**2 + th[1]*x2**2 + th[2]
20

```

```

21
22 def f_vector_(x_vector):
23     x1 = float(x_vector[0])
24     x2 = float(x_vector[1])
25     return [x1**2, x2**2, 1]
26
27
28 def f_vector(th, x_vector):
29     x1 = float(x_vector[0])
30     x2 = float(x_vector[1])
31     return [th[0]*x1**2, th[1]*x2**2, th[2]]

```

Файл model_4parameters.py

```

1 # файл параметров модели
2
3 theta_true = [0.1, -1, 0.1, 3] # параметры уравнения  $t_0*x_1^2 + t_1*x_2^2 + t_2*x_1 + t_3$ 
4 a, b, n, m = -1, 1, 100, 4 # границы, число точек и параметров
5 rho = 0.15 # шум в диапазоне [0.05, 0.15] или [0.50, 0.70]
6 alpha = 0.05 # уровень значимости
7 dist_E = 0.000491836210109302 # для theta_true = [-0.05, 2, 4]
8 f_E = 9000 # вместо float('inf')
9 count = 11
10
11
12 def f_2_parameters(th, x1, x2):
13     return th[0]*x1**2 + th[1]*x2**2 + th[2]*x1 + th[3]
14
15
16 def f(th, x_vector):
17     x1 = float(x_vector[0])
18     x2 = float(x_vector[1])
19     return th[0]*x1**2 + th[1]*x2**2 + th[2]*x1 + th[3]
20
21
22 def f_vector_(x_vector):
23     x1 = float(x_vector[0])
24     x2 = float(x_vector[1])
25     return [x1**2, x2**2, x1, 1]
26
27
28 def f_vector(th, x_vector):
29     x1 = float(x_vector[0])
30     x2 = float(x_vector[1])
31     return [th[0]*x1**2, th[1]*x2**2, th[2]*x1, th[3]]

```

Файл lab1.py

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3 from mpl_toolkits.mplot3d import Axes3D
4 from scipy import stats
5 from sympy import *
6 from sys import argv, path
7
8
9 #
10 #
11 #
12 class lab1():

```

```

13 def __init__(self):
14     self.x = np.ndarray((n, 2))
15     self.x1 = np.random.uniform(low=a, high=b, size=n)
16     self.x2 = np.random.uniform(low=a, high=b, size=n)
17     self.u = np.ndarray(n)
18     for i in range(n):
19         self.x[i] = [self.x1[i], self.x2[i]]
20     for i in range(n):
21         self.u[i] = f(theta_true, self.x[i])
22     u_mean = np.full((n), np.mean(self.u))
23     w_squared = np.dot((self.u - u_mean), (self.u - u_mean)) / (n-1)
24     self.dist = rho * w_squared
25     self.y = np.copy(self.u)
26     for i in range(n):
27         self.y[i] += np.random.normal(0, self.dist)
28
29 def save_table_1(self):
30     with open('1/report/table_1_u_y_' + str(int(rho*100)) + '.txt', 'w') as
file:
31         file.write('i\tx1\tx2\tu\ty\n')
32         for i in range(n):
33             x1 = self.x1[i]
34             x2 = self.x2[i]
35             file.write('{:d}\t{:.17f}\t{:.17f}\t{:.17f}\t{:.17f}\n'.format(
36                 i, x1, x2, self.u[i], self.y[i]))
37
38 def draw(self, doDrawWithNoize, title, name, elevation, azimuth):
39     path_to_save = '1/pics/' + name + '_' + \
40         str(int(rho*100)) + '_' + str(elevation) + '_' + str(azimuth) + '.
png'
41     fig = plt.figure('1')
42     ax = fig.gca(projection='3d')
43     tmp_range = np.linspace(a, b, sqrt(n))
44     x1, x2 = np.meshgrid(tmp_range, tmp_range)
45     u = f_2_parameters(theta_true, x1, x2)
46     fig.subplots_adjust(bottom=-0.05, top=1, left=-0.05, right=1.05)
47     ax.view_init(elevation, azimuth)
48     ax.plot_surface(x1, x2, u, zorder=2, alpha=0.2)
49     if doDrawWithNoize:
50         title += ' ' + str(int(rho * 100)) + '%'
51         ax.scatter(self.x1, self.x2, self.y, c='black', zorder=1)
52     else:
53         ax.scatter(self.x1, self.x2, self.u, c='black', zorder=1)
54     plt.title(title, fontsize=19)
55     plt.xlabel('x1')
56     plt.ylabel('x2')
57     plt.grid(alpha=0.5)
58     plt.savefig(path_to_save)
59     plt.clf()
60
61 #
62 #
63 #
64 #
65 if __name__ == "__main__":
66     path.insert(1, '')
67     if int(argv[2]) == 3:
68         from model_3parameters import *
69     else:
70         from model_4parameters import *

```

```

71 global rho
72 rho = float(argv[1])
73 print('\nЗапущен код 1 лабораторной работы: {:d} параметров, шум {:d}%'.format
(int(argv[2]), int(rho*100)))
74
75 l1 = lab1()
76 l1.save_table_1()
77 l1.draw(False, 'исходная модель', 'before', None, None)
78 l1.draw(False, 'исходная модель', 'before', 0, 0)
79 l1.draw(False, 'исходная модель', 'before', 0, 90)
80 l1.draw(True, 'помеха', 'after', None, None)
81 l1.draw(True, 'помеха', 'after', 0, 0)
82 l1.draw(True, 'помеха', 'after', 0, 90)

```

Файл lab2.py

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3 from mpl_toolkits.mplot3d import Axes3D
4 from scipy import stats
5 from sympy import *
6 from sys import argv, path
7
8
9 #
10 #
11 #
12 class lab2():
13     def __init__(self):
14         self.x = np.ndarray((n, 2))
15         self.x1 = np.ndarray(n)
16         self.x2 = np.ndarray(n)
17         self.u = np.ndarray(n)
18         self.y = np.ndarray(n)
19         with open('1/report/table_1_u_y_' + str(int(rho*100)) + '.txt', 'r') as
file:
20             line = file.readline().rstrip()
21             for i in range(n):
22                 line = file.readline().rstrip().rsplit()
23                 self.x1[i] = float(line[1])
24                 self.x2[i] = float(line[2])
25                 self.u[i] = float(line[3])
26                 self.y[i] = float(line[4])
27
28             for i in range(n):
29                 self.x[i] = [self.x1[i], self.x2[i]]
30
31             self.X = np.ndarray((n, m))
32             for i in range(n):
33                 self.X[i] = f_vector_(self.x[i])
34
35     def calc_theta(self):
36         X = Matrix(self.X)
37         y = Matrix(self.y)
38         theta_calc = ((X.T * X) ** -1) * X.T * y
39         self.theta_calc = np.array(theta_calc.T).astype(np.float64)[0]
40
41     def calc_e(self):
42         X = Matrix(self.X)
43         y = Matrix(self.y)
44         th_calc = Matrix(self.theta_calc)

```

```

45     yCalc = X * th_calc
46     e_calc = y - yCalc
47     self.yCalc = np.array(yCalc.T).astype(np.float64)[0]
48     self.e_calc = np.array(e_calc.T).astype(np.float64)[0]
49
50     def calc_dist(self):
51         e_calc = Matrix(self.e_calc)
52         dist_calc = (e_calc.T * e_calc) / (n - m)
53         self.dist_calc = np.array(dist_calc.T).astype(np.float64)[0][0]
54
55     def draw(self, title, name, elevation, azimuth):
56         path_to_save = '2/pics/' + name + '_' + str(int(rho*100)) + '_' + str(
elevation) + '_' + str(azimuth) + '.png'
57         title += ' ' + str(int(rho * 100)) + '%'
58         fig = plt.figure('1')
59         ax = fig.gca(projection='3d')
60         _t = np.linspace(a, b, sqrt(n))
61         _x, _y = np.meshgrid(_t, _t)
62         _z = f_2_parameters(theta_true, _x, _y)
63         fig.subplots_adjust(bottom=-0.05, top=1, left=-0.05, right=1.05)
64         ax.view_init(elevation, azimuth)
65         ax.plot_surface(_x, _y, _z, zorder=2, alpha=0.2)
66         ax.scatter(self.x1, self.x2, self.yCalc, c='black')
67         plt.title(title, fontsize=19)
68         plt.xlabel('X')
69         plt.ylabel('Y')
70         plt.grid(alpha=0.4)
71         plt.savefig(path_to_save)
72         plt.clf()
73
74     def check_goodness_of_fit(self):
75         self.F = self.dist_calc / dist_E
76         d1, d2 = n-m, f_E
77         self.F_t = stats.f.ppf(1 - alpha, d1, d2)
78         print('F: ' + str(self.F) + ' F_t ' + str(self.F_t))
79         if self.F <= self.F_t:
80             print("Гипотеза не отвергается")
81         else:
82             print("Полученная модель неадекватна")
83
84     def save_table_2(self):
85         with open('2/report/table_2_u_y_' + str(int(rho*100)) + '.txt', 'w') as
file:
86             file.write('i\tx1\tx2\tu\tyCalc\tyMinyCalc\n')
87             for i in range(n):
88                 x1 = self.x1[i]
89                 x2 = self.x2[i]
90                 u = self.u[i]
91                 y = self.y[i]
92                 y_minus_yCalc = float(self.y[i] - self.u[i])
93                 file.write('{:d}\t{:.17f}\t{:.17f}\t{:.17f}\t{:.17f}\t{:.17f}\n'
.format(i, x1, x2, u, y, y_minus_yCalc))
94
95     def save_table_3(self):
96         with open('2/report/table_3_th_' + str(int(rho*100)) + '.txt', 'w') as
file:
97             file.write('theta_true\ttheta_calc\n')
98             for i in range(m):
99                 file.write('{:.17f}\t{:.17f}\n'.format(theta_true[i], self.
theta_calc[i]))

```



```

100
101     def save_table_4(self):
102         with open('2/report/table_4_dist_F_' + str(int(rho*100)) + '.txt', 'w')
as file:
103             file.write('dist_calc\tdist_E\tF\tF_t\n')
104             file.write('{:.17f}\t{:.17f}\t{:.17f}\t{:.17f}\n'.format(self.
dist_calc, dist_E, self.F, self.F_t))
105
106
107
108
109 #
110 #
111 #
112 if __name__ == "__main__":
113     path.insert(1, '')
114     if int(argv[2]) == 3:
115         from model_3parameters import *
116     else:
117         from model_4parameters import *
118     global rho
119     rho = float(argv[1])
120     print('\nЗапущен код 2 лабораторной работы: {:d} параметров, шум {:d}%'.format
(int(argv[2]), int(rho*100)))
121
122     l2 = lab2()
123     l2.calc_theta()
124     l2.calc_e()
125     l2.calc_dist()
126     l2.draw('полученная модель', 'calculated', None, None)
127     l2.draw('полученная модель', 'calculated', 0, 0)
128     l2.draw('полученная модель', 'calculated', 0, 90)
129     l2.check_goodness_of_fit()
130     l2.save_table_2()
131     l2.save_table_3()
132     l2.save_table_4()

```

Файл lab3.py

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3 from mpl_toolkits.mplot3d import Axes3D
4 from scipy import stats
5 from sympy import *
6 from sys import argv, path
7
8
9 #
10 #
11 #
12 class lab3():
13     def __init__(self):
14         self.x = np.ndarray((n, 2))
15         self.x1 = np.ndarray(n)
16         self.x2 = np.ndarray(n)
17         self.u = np.ndarray(n)
18         self.y = np.ndarray(n)
19         with open('1/report/table_1_u_y_' + str(int(rho*100)) + '.txt', 'r') as
file:
20             line = file.readline().rstrip()
21             for i in range(n):

```

```

22         line = file.readline().rstrip().rsplit()
23         self.x1[i] = float(line[1])
24         self.x2[i] = float(line[2])
25         self.u[i] = float(line[3])
26         self.y[i] = float(line[4])
27
28     for i in range(n):
29         self.x[i] = [self.x1[i], self.x2[i]]
30
31     self.X = np.ndarray((n, m))
32     for i in range(n):
33         self.X[i] = f_vector_(self.x[i])
34
35     global theta_true
36     self.theta_calc = np.ndarray(m)
37     with open('2/report/table_3_th_' + str(int(rho*100)) + '.txt', 'r') as
file:
38         line = file.readline()
39         for i in range(m):
40             line = file.readline().rstrip().rsplit()
41             theta_true[i] = line[0]
42             self.theta_calc[i] = line[1]
43
44     with open('2/report/table_4_dist_F_' + str(int(rho*100)) + '.txt', 'r')
as file:
45         line = file.readline()
46         data = [float(_) for _ in file.readline().rstrip().rsplit()]
47         self.dist_calc = data[0]
48         dist_E = data[1]
49         self.F = data[2]
50         self.F_t = data[3]
51
52     def build_conf_interval_theta(self):
53         X = Matrix(self.X)
54         self.matrix = ((X.T*X)**-1).tolist()
55         self.t = abs(stats.t.ppf(1 - alpha/2, n - m))
56         self.thera_lower = np.ndarray(m)
57         self.thera_upper = np.ndarray(m)
58         for i in range(m):
59             param = self.theta_calc[i]
60             self.thera_lower[i] = param - self.t * self.matrix[i][i]
61             self.thera_upper[i] = param + self.t * self.matrix[i][i]
62
63     def check_significance_of_parameters(self):
64         self.F_t = stats.f.ppf(1 - alpha, 1, n - m)
65         self.F = np.ndarray(m)
66         for i in range(m):
67             self.F[i] = self.theta_calc[i]**2 / \
68                 (self.dist_calc * self.matrix[i][i])
69
70     def save_table_4(self):
71         with open('3/report/table_4_th_' + str(int(rho*100)) + '.txt', 'w') as
file:
72             file.write('theta_l\ttheta\ttheta_r\tF\tF_t\tsignificance\n')
73             for i in range(m):
74                 file.write('{:.17f}\t{:.17f}\t{:.17f}\t{:.17f}\t{:.17f}\t'.
format(
75                     self.thera_lower[i],
76                     self.theta_calc[i],
77                     self.thera_upper[i],

```

```

78         self.F[i],
79         self.F_t
80     ))
81     if self.F[i] < self.F_t:
82         file.write('-\n')
83     else:
84         file.write('+ \n')
85
86 def check_significance_of_regression(self):
87     y = Matrix(self.y)
88     X = Matrix(self.X)
89     th = Matrix(self.theta_calc)
90     RSS = (y - X*th).T * (y - X*th)
91     self.RSS = np.array(RSS).astype(np.float64)[0][0]
92
93     self.RSS_H = 0
94     yMean = np.full((n), np.mean(self.y))
95     for i in range(n):
96         self.RSS_H += (self.y[i] - yMean[i])**2
97
98     q = m - 1
99     self.regr_F = ((self.RSS_H - self.RSS) / q) / (self.RSS / (n-m))
100    self.regr_F_t = stats.f.ppf(1 - alpha, q, n - m)
101    if self.regr_F <= self.regr_F_t:
102        print('Гипотеза о незначимости регрессии принимается {:.f} <= {:.f}'.
format(self.regr_F, self.regr_F_t))
103    else:
104        print('Гипотеза о незначимости регрессии отвергается {:.f} > {:.f}'.format
(self.regr_F, self.regr_F_t))
105
106 def estimate_E_for_x1(self):
107     path_to_save = '3/pics/conf_y_for_x1_' + str(int(rho*100)) + '.png'
108     x1_list = np.linspace(a, b, count)
109     x2 = 0 # середина D(f)
110     X = Matrix(self.X)
111     lower = np.ndarray(count)
112     center = np.ndarray(count)
113     upper = np.ndarray(count)
114     for i in range(count):
115         x1 = x1_list[i]
116         x = [x1, x2]
117         fx = f(self.theta_calc, x)
118         fx_vector = Matrix(f_vector(self.theta_calc, x))
119         tmp = fx_vector.T * (X.T * X)**-1 * fx_vector
120         sigma = self.dist_calc * \
121             (1 + np.array(tmp).astype(np.float64)[0][0])
122         lower[i] = fx - self.t * sigma
123         center[i] = fx
124         upper[i] = fx + self.t * sigma
125
126     plt.title('оценка M(x1), помеха ' + str(int(rho * 100)) + '%')
127     plt.plot(lower)
128     plt.plot(center)
129     plt.plot(upper)
130     plt.savefig(path_to_save)
131     plt.clf()
132
133 def estimate_E_for_x2(self):
134     path_to_save = '3/pics/conf_y_for_x2_' + str(int(rho*100)) + '.png'
135     x1 = 0 # середина D(f)

```

```

136     x2_list = np.linspace(a, b, count)
137     X = Matrix(self.X)
138     lower = np.ndarray(count)
139     center = np.ndarray(count)
140     upper = np.ndarray(count)
141     for i in range(count):
142         x2 = x2_list[i]
143         x = [x1, x2]
144         fx = f(self.theta_calc, x)
145         fx1vector = Matrix(f_vector(self.theta_calc, x))
146         tmp = sqrt(fx1vector.T * (X.T * X)**-1 * fx1vector)
147         sigma = sqrt(self.dist_calc) * \
148             np.array(tmp).astype(np.float64)[0][0]
149         lower[i] = fx - self.t * sigma
150         center[i] = fx
151         upper[i] = fx + self.t * sigma
152
153     plt.title('оценка M(x2), помеха ' + str(int(rho * 100)) + '%')
154     plt.plot(lower)
155     plt.plot(center)
156     plt.plot(upper)
157     plt.savefig(path_to_save)
158     plt.clf()
159
160
161 #
162 #
163 #
164 if __name__ == "__main__":
165     path.insert(1, '')
166     if int(argv[2]) == 3:
167         from model_3parameters import *
168     else:
169         from model_4parameters import *
170     global rho
171     rho = float(argv[1])
172     print('\nЗапущен код 3 лабораторной работы: {:d} параметров, шум {:d}%'.format
173           (int(argv[2]), int(rho*100)))
174
175     l3 = lab3()
176     l3.build_conf_interval_theta()
177     l3.check_significance_of_parameters()
178     l3.save_table_4()
179     l3.check_significance_of_regression()
180     l3.estimate_E_for_x1()
181     l3.estimate_E_for_x2()

```