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

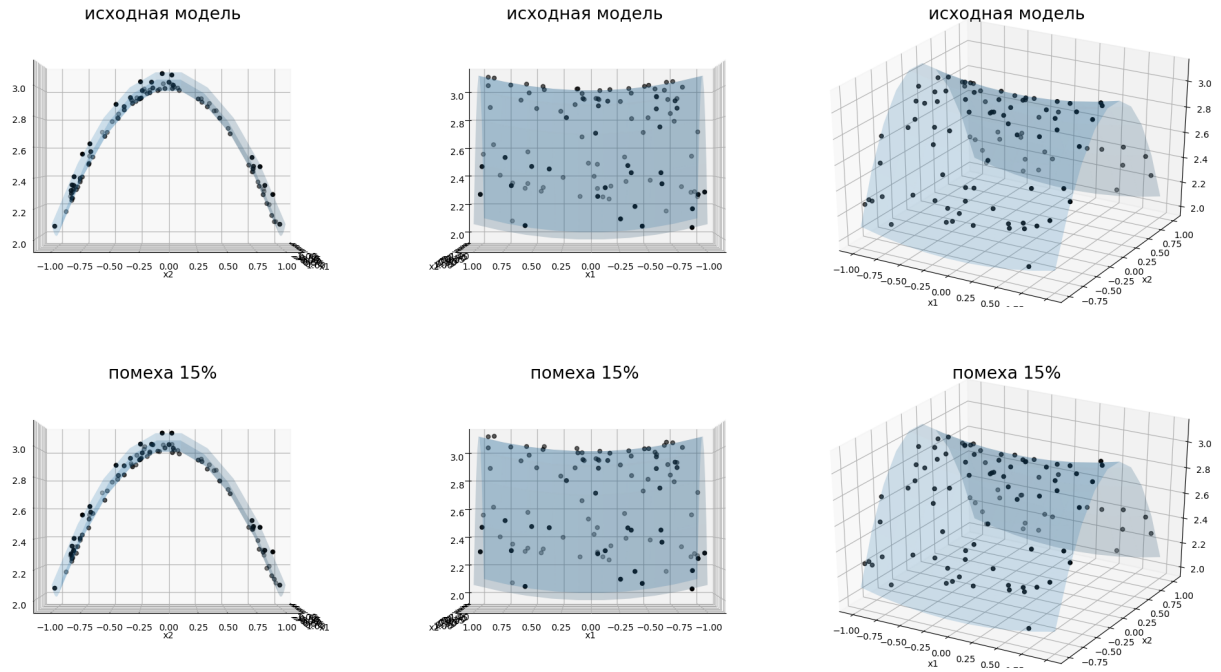
Новосибирск
2019

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 расмотрении 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	$\mathrm{F}_{\alpha,q,n-m}$	значимость параметра
-0.115	0.103	0.321	748.312	3.940	$+$
-1.209	-1.005	-0.800	75,948.790	3.940	$+$
-0.061	-0.001	0.058	0.418	3.940	-
2.937	3.001	3.065	2,161,261.997	3.940	$+$

Как мы видим новый регрессор оказался незначим.

4. Проверить гипотезу о незначимости самой регрессии

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

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

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

$$q = m - 1$$

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

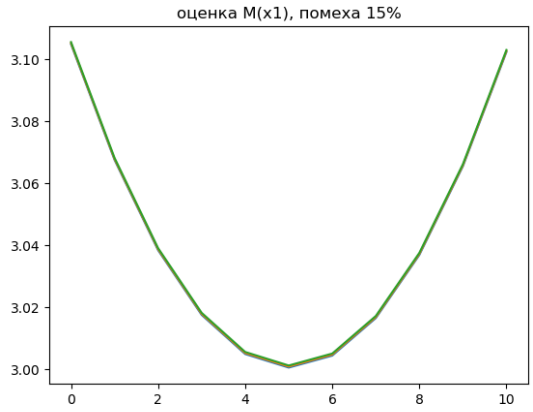
5. Рассчитать прогнозные занчения для математического ожидания функции отклика $\eta(x,\hat{\theta}) = f^T(x)\hat{\theta}$ для всего интервала действия одного из факторов, зафиксировав значения других факторов на границе или в центре области их определения.

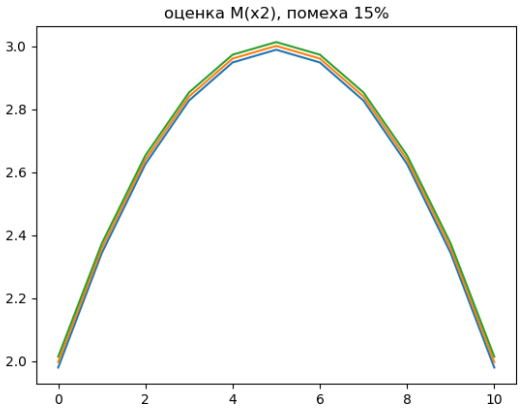
6. По полученным в п.5 прогнозным значениям построить графики прогнозных значений и доверительной полосы для математического ожидания функцииspa отклика и для самого отклика.

Для оценки математического ожидания функции отклика построим доверительный интервал:

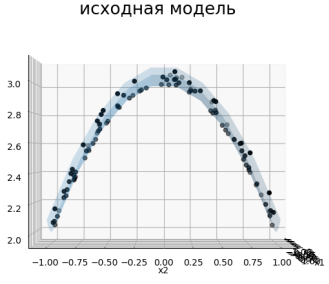
$$\eta(x,\hat{\theta}) - t_{/2,f_R}\sigma(\eta(x,\hat{\theta})) \le \eta(x,\hat{\theta}) \le \eta(x,\hat{\theta}) + t_{/2,f_R}\sigma(\eta(x,\hat{\theta}))$$

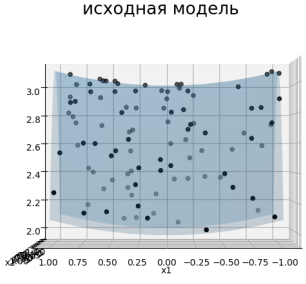
$$\sigma(\eta(x,\hat{\theta})) = \hat{\sigma}\sqrt{1 + f^T(x)(X^TX)^{-1}f(x)}$$

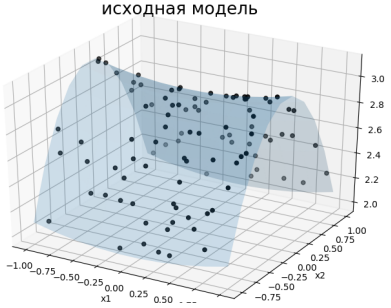


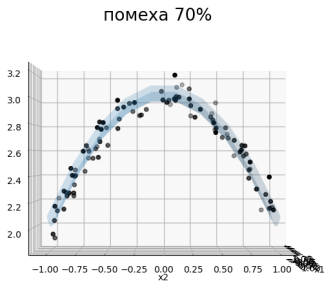


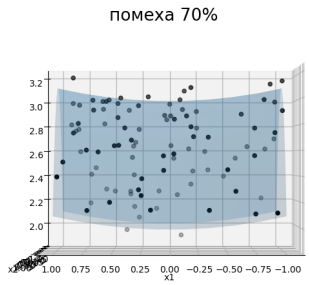
7. Заново смоделировать исходные данные (см. лаб. работу №1), увеличив мощность случайной помехи до 50...70% от мощности полезного сигнала, и провести оценку параметров. Повторить пункты 3, 4 с новыми данными.

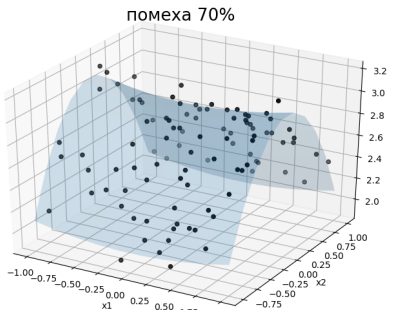












$\hat{\theta}_j - t_{/2,f_R}\sigma(\hat{\sigma}_j)$	$\hat{\theta}_j$	$\hat{\theta}_j + t_{/2,f_R}\sigma(\hat{\sigma}_j)$	F	$\mathrm{F}_{\alpha,q,n-m}$	значимость параметра
-0.111	0.111	0.334	23.717	3.940	+
-1.215	-0.993	-0.771	1,889.521	3.940	+
-0.071	-0.004	0.062	0.127	3.940	-
2.912	2.993	3.074	46,930.283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

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

3. Исходный код программы

Файл run_lab3.py

```python
import os

os.system('python 1/lab1.py 0.15 3')
os.system('python 2/lab2.py 0.15 4')
os.system('python 3/lab3.py 0.15 4')

os.system('python 1/lab1.py 0.70 3')
os.system('python 2/lab2.py 0.70 4')
os.system('python 3/lab3.py 0.70 4')
```

Файл model_3parameters.py

```python
# файл параметров модели

theta_true = [0.1, -1, 3]        # параметры уравнения t0*x1^2 + t1*x2^2 + t2
a, b, n, m = -1, 1, 100, 3       # границы, число точек и параметров
rho = 0.15                       # шум в диапазоне [0.05, 0.15] или [0.50, 0.70]
alpha = 0.05                     # уровень значимости
dist_E = 0.000491836210109302    # для theta_true = [-0.05, 2, 4]
f_E = 9000                       # вместо float('inf')
count = 11


def f_2_parameters(th, x1, x2):
    return th[0]*x1**2 + th[1]*x2**2 + th[2]


def f(th, x_vector):
    x1 = float(x_vector[0])
    x2 = float(x_vector[1])
    return th[0]*x1**2 + th[1]*x2**2 + th[2]

```

```
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 t0*x1^2 + t1*x2^2 + th2*x1 +
       t3
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():
```

```
    def __init__(self):
        self.x = np.ndarray((n, 2))
        self.x1 = np.random.uniform(low=a, high=b, size=n)
        self.x2 = np.random.uniform(low=a, high=b, size=n)
        self.u = np.ndarray(n)
        for i in range(n):
            self.x[i] = [self.x1[i], self.x2[i]]
        for i in range(n):
            self.u[i] = f(theta_true, self.x[i])
        u_mean = np.full((n), np.mean(self.u))
        w_squared = np.dot((self.u - u_mean), (self.u - u_mean)) / (n-1)
        self.dist = rho * w_squared
        self.y = np.copy(self.u)
        for i in range(n):
            self.y[i] += np.random.normal(0, self.dist)

    def save_table_1(self):
        with open('1/report/table_1_u_y_' + str(int(rho*100)) + '.txt', 'w') as file:
            file.write('i\tx1\tx2\tu\ty\n')
            for i in range(n):
                x1 = self.x1[i]
                x2 = self.x2[i]
                file.write('{:d}\t{:.17f}\t{:.17f}\t{:.17f}\t{:.17f}\n'.format(
                    i, x1, x2, self.u[i], self.y[i]))

    def draw(self, doDrawWithNoize, title, name, elevation, azimuth):
        path_to_save = '1/pics/' + name + '_' + \
            str(int(rho*100)) + '_' + str(elevation) + '_' + str(azimuth) + '.png'
        fig = plt.figure('1')
        ax = fig.gca(projection='3d')
        tmp_range = np.linspace(a, b, sqrt(n))
        x1, x2 = np.meshgrid(tmp_range, tmp_range)
        u = f_2_parameters(theta_true, x1, x2)
        fig.subplots_adjust(bottom=-0.05, top=1, left=-0.05, right=1.05)
        ax.view_init(elevation, azimuth)
        ax.plot_surface(x1, x2, u, zorder=2, alpha=0.2)
        if doDrawWithNoize:
            title += ' ' + str(int(rho * 100)) + '%'
            ax.scatter(self.x1, self.x2, self.y, c='black', zorder=1)
        else:
            ax.scatter(self.x1, self.x2, self.u, c='black', zorder=1)
        plt.title(title, fontsize=19)
        plt.xlabel('x1')
        plt.ylabel('x2')
        plt.grid(alpha=0.5)
        plt.savefig(path_to_save)
        plt.clf()


#-------------------------------------------------------------------------------
#-------------------------------------------------------------------------------
#-------------------------------------------------------------------------------
if __name__ == "__main__":
    path.insert(1, '')
    if int(argv[2]) == 3:
        from model_3parameters import *
    else:
        from model_4parameters import *
```

```
    global rho
    rho = float(argv[1])
    print('\n\Запущенn код 1 лабораторной работы: {:d} параметров, шум {:d}%'.format
    (int(argv[2]), int(rho*100)))

    l1 = lab1()
    l1.save_table_1()
    l1.draw(False, 'исходная модель', 'before', None, None)
    l1.draw(False, 'исходная модель', 'before', 0, 0)
    l1.draw(False, 'исходная модель', 'before', 0, 90)
    l1.draw(True, 'помеха', 'after', None, None)
    l1.draw(True, 'помеха', 'after', 0, 0)
    l1.draw(True, 'помеха', 'after', 0, 90)
```

Файл lab2.py

```
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
from scipy import stats
from sympy import *
from sys import argv, path


#-----------------------------------------------------------------------
#-----------------------------------------------------------------------
#-----------------------------------------------------------------------
class lab2():
    def __init__(self):
        self.x = np.ndarray((n, 2))
        self.x1 = np.ndarray(n)
        self.x2 = np.ndarray(n)
        self.u = np.ndarray(n)
        self.y = np.ndarray(n)
        with open('1/report/table_1_u_y_' + str(int(rho*100)) + '.txt', 'r') as
    file:
            line = file.readline().rstrip()
            for i in range(n):
                line = file.readline().rstrip().rsplit()
                self.x1[i] = float(line[1])
                self.x2[i] = float(line[2])
                self.u[i] = float(line[3])
                self.y[i] = float(line[4])

        for i in range(n):
            self.x[i] = [self.x1[i], self.x2[i]]

        self.X = np.ndarray((n, m))
        for i in range(n):
            self.X[i] = f_vector_(self.x[i])

    def calc_theta(self):
        X = Matrix(self.X)
        y = Matrix(self.y)
        theta_calc = ((X.T * X) ** -1) * X.T * y
        self.theta_calc = np.array(theta_calc.T).astype(np.float64)[0]

    def calc_e(self):
        X = Matrix(self.X)
        y = Matrix(self.y)
        th_calc = Matrix(self.theta_calc)
```

```
        yCalc = X * th_calc
        e_calc = y - yCalc
        self.yCalc = np.array(yCalc.T).astype(np.float64)[0]
        self.e_calc = np.array(e_calc.T).astype(np.float64)[0]

    def calc_dist(self):
        e_calc = Matrix(self.e_calc)
        dist_calc = (e_calc.T * e_calc) / (n - m)
        self.dist_calc = np.array(dist_calc.T).astype(np.float64)[0][0]

    def draw(self, title, name, elevation, azimuth):
        path_to_save = '2/pics/' + name + '_' + str(int(rho*100)) + '_' + str(elevation) + '_' + str(azimuth) + '.png'
        title += ' ' + str(int(rho * 100)) + '%'
        fig = plt.figure('1')
        ax = fig.gca(projection='3d')
        _t = np.linspace(a, b, sqrt(n))
        _x, _y = np.meshgrid(_t, _t)
        _z = f_2_parameters(theta_true, _x, _y)
        fig.subplots_adjust(bottom=-0.05, top=1, left=-0.05, right=1.05)
        ax.view_init(elevation, azimuth)
        ax.plot_surface(_x, _y, _z, zorder=2, alpha=0.2)
        ax.scatter(self.x1, self.x2, self.yCalc, c='black')
        plt.title(title, fontsize=19)
        plt.xlabel('X')
        plt.ylabel('Y')
        plt.grid(alpha=0.4)
        plt.savefig(path_to_save)
        plt.clf()

    def check_goodness_of_fit(self):
        self.F = self.dist_calc / dist_E
        d1, d2 = n-m, f_E
        self.F_t = stats.f.ppf(1 - alpha, d1, d2 )
        print('F: ' + str(self.F) + ' F_t ' + str(self.F_t))
        if self.F <= self.F_t:
            print("Гипотеза не отвергается")
        else:
            print("Полученная модель неадекватна")

    def save_table_2(self):
        with open('2/report/table_2_u_y_' + str(int(rho*100)) + '.txt', 'w') as file:
            file.write('i\tx1\tx2\tu\tyCalc\tyMinyCalc\n')
            for i in range(n):
                x1 = self.x1[i]
                x2 = self.x2[i]
                u = self.u[i]
                y = self.y[i]
                y_minus_yCalc = float(self.y[i] - self.u[i])
                file.write('{:d}\t{:.17f}\t{:.17f}\t{:.17f}\t{:.17f}\t{:.17f}\n'.format(i, x1, x2, u, y, y_minus_yCalc))

    def save_table_3(self):
        with open('2/report/table_3_th_' + str(int(rho*100)) + '.txt', 'w') as file:
            file.write('theta_true\ttheta_calc\n')
            for i in range(m):
                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('\n\Запущенn 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):
```

```
                line = file.readline().rstrip().rsplit()
                self.x1[i] = float(line[1])
                self.x2[i] = float(line[2])
                self.u[i] = float(line[3])
                self.y[i] = float(line[4])

        for i in range(n):
            self.x[i] = [self.x1[i], self.x2[i]]

        self.X = np.ndarray((n, m))
        for i in range(n):
            self.X[i] = f_vector_(self.x[i])

        global theta_true
        self.theta_calc = np.ndarray(m)
        with open('2/report/table_3_th_' + str(int(rho*100)) + '.txt', 'r') as
    file:
            line = file.readline()
            for i in range(m):
                line = file.readline().rstrip().rsplit()
                theta_true[i] = line[0]
                self.theta_calc[i] = line[1]

        with open('2/report/table_4_dist_F_' + str(int(rho*100)) + '.txt', 'r')
    as file:
            line = file.readline()
            data = [float(_) for _ in file.readline().rstrip().rsplit()]
            self.dist_calc = data[0]
            dist_E = data[1]
            self.F = data[2]
            self.F_t = data[3]

    def build_conf_interval_theta(self):
        X = Matrix(self.X)
        self.matrix = ((X.T*X)**-1).tolist()
        self.t = abs(stats.t.ppf(1 - alpha/2, n - m))
        self.thera_lower = np.ndarray(m)
        self.thera_upper = np.ndarray(m)
        for i in range(m):
            param = self.theta_calc[i]
            self.thera_lower[i] = param - self.t * self.matrix[i][i]
            self.thera_upper[i] = param + self.t * self.matrix[i][i]

    def check_significance_of_parameters(self):
        self.F_t = stats.f.ppf(1 - alpha, 1, n - m)
        self.F = np.ndarray(m)
        for i in range(m):
            self.F[i] = self.theta_calc[i]**2 / \
                (self.dist_calc * self.matrix[i][i])

    def save_table_4(self):
        with open('3/report/table_4_th_' + str(int(rho*100)) + '.txt', 'w') as
    file:
            file.write('theta_l\ttheta\ttheta_r\tF\tF_t\tsignificance\n')
            for i in range(m):
                file.write('{:.17f}\t{:.17f}\t{:.17f}\t{:.17f}\t{:.17f}\t'.
    format(
                    self.thera_lower[i],
                    self.theta_calc[i],
                    self.thera_upper[i],
```

```
                    self.F[i],
                    self.F_t
                ))
                if self.F[i] < self.F_t:
                    file.write('-\n')
                else:
                    file.write('+\n')

    def check_significance_of_regression(self):
        y = Matrix(self.y)
        X = Matrix(self.X)
        th = Matrix(self.theta_calc)
        RSS = (y - X*th).T * (y - X*th)
        self.RSS = np.array(RSS).astype(np.float64)[0][0]

        self.RSS_H = 0
        yMean = np.full((n), np.mean(self.y))
        for i in range(n):
            self.RSS_H += (self.y[i] - yMean[i])**2

        q = m - 1
        self.regr_F = ((self.RSS_H - self.RSS) / q) / (self.RSS / (n-m))
        self.regr_F_t = stats.f.ppf(1 - alpha, q, n - m)
        if self.regr_F <= self.regr_F_t:
            print('Гипотеза о незначимости регрессии принимается {:f} <= {:f}'.
    format(self.regr_F, self.regr_F_t))
        else:
            print('Гипотеза о незначимости регрессии отвергается {:f} > {:f}'.format
    (self.regr_F, self.regr_F_t))

    def estimate_E_for_x1(self):
        path_to_save = '3/pics/conf_y_for_x1_' + str(int(rho*100)) + '.png'
        x1_list = np.linspace(a, b, count)
        x2 = 0  # середина D(f)
        X = Matrix(self.X)
        lower = np.ndarray(count)
        center = np.ndarray(count)
        upper = np.ndarray(count)
        for i in range(count):
            x1 = x1_list[i]
            x = [x1, x2]
            fx = f(self.theta_calc, x)
            fx_vector = Matrix(f_vector(self.theta_calc, x))
            tmp = fx_vector.T * (X.T * X)**-1 * fx_vector
            sigma = self.dist_calc * \
                (1 + np.array(tmp).astype(np.float64)[0][0])
            lower[i] = fx - self.t * sigma
            center[i] = fx
            upper[i] = fx + self.t * sigma

        plt.title('оценка M(x1), помеха ' + str(int(rho * 100)) + '%')
        plt.plot(lower)
        plt.plot(center)
        plt.plot(upper)
        plt.savefig(path_to_save)
        plt.clf()

    def estimate_E_for_x2(self):
        path_to_save = '3/pics/conf_y_for_x2_' + str(int(rho*100)) + '.png'
        x1 = 0  # середина D(f)
```

```
        x2_list = np.linspace(a, b, count)
        X = Matrix(self.X)
        lower = np.ndarray(count)
        center = np.ndarray(count)
        upper = np.ndarray(count)
        for i in range(count):
            x2 = x2_list[i]
            x = [x1, x2]
            fx = f(self.theta_calc, x)
            fx1vector = Matrix(f_vector(self.theta_calc, x))
            tmp = sqrt(fx1vector.T * (X.T * X)**-1 * fx1vector)
            sigma = sqrt(self.dist_calc) * \
                np.array(tmp).astype(np.float64)[0][0]
            lower[i] = fx - self.t * sigma
            center[i] = fx
            upper[i] = fx + self.t * sigma


        plt.title('оценка M(x2), помеха ' + str(int(rho * 100)) + '%')
        plt.plot(lower)
        plt.plot(center)
        plt.plot(upper)
        plt.savefig(path_to_save)
        plt.clf()


#-----------------------------------------------------------------------
#-----------------------------------------------------------------------
#-----------------------------------------------------------------------
if __name__ == "__main__":
    path.insert(1, '')
    if int(argv[2]) == 3:
        from model_3parameters import *
    else:
        from model_4parameters import *
    global rho
    rho = float(argv[1])
    print('\n\Запушенn код 3 лабораторной работы: {:d} параметров, шум {:d}%'.format
    (int(argv[2]), int(rho*100)))

    l3 = lab3()
    l3.build_conf_interval_theta()
    l3.check_significance_of_parameters()
    l3.save_table_4()
    l3.check_significance_of_regression()
    l3.estimate_E_for_x1()
    l3.estimate_E_for_x2()
```