Лабораторная работа №8. Вариант 11.

Курбатов Егор Павлович M3238 exxxcilus@niuitmo.ru

13 июня 2020 г.

1 Исходные данные

$x_{min} = -1.3$, $x_{max} = 1.7$ — границы интервала.
$n = 40$ — количество точек.
$a_1 = 1.1$, $a_2 = -2.2$, $a_3 = 3.7$ — коэффициенты квадратичной функции.
$c_1 = 1.1$, $c_2 = 2.5$ — коэффициенты линейной функции.
$s = 2.7$ — уровень шумов.

2 Задание

1. (a) Смоделировать квадратичную функцию, наблюдаемую в нормальных шумах в пакете Octave в соответствии с параметрами варианта.
   (b) Оценить коэффициенты квадратичной зависимости, уровень шумов и квадратичную функцию по зашумленным данным.
   (c) Сравнить полученные результаты с "истинными" данными.
2. (a) Смоделировать линейную функцию, наблюдаемую в нормальных шумах в пакете Octave в соответствии с параметрами варианта.
   (b) Оценить коэффициенты линейной зависимости, уровень шумов и линейную функцию по зашумленным данным.
   (c) Построить доверительный интервал для значений функции при уровне доверии 0.95.
   (d) Сравнить полученные результаты с "истинными" данными.

3 Квадратичная функция

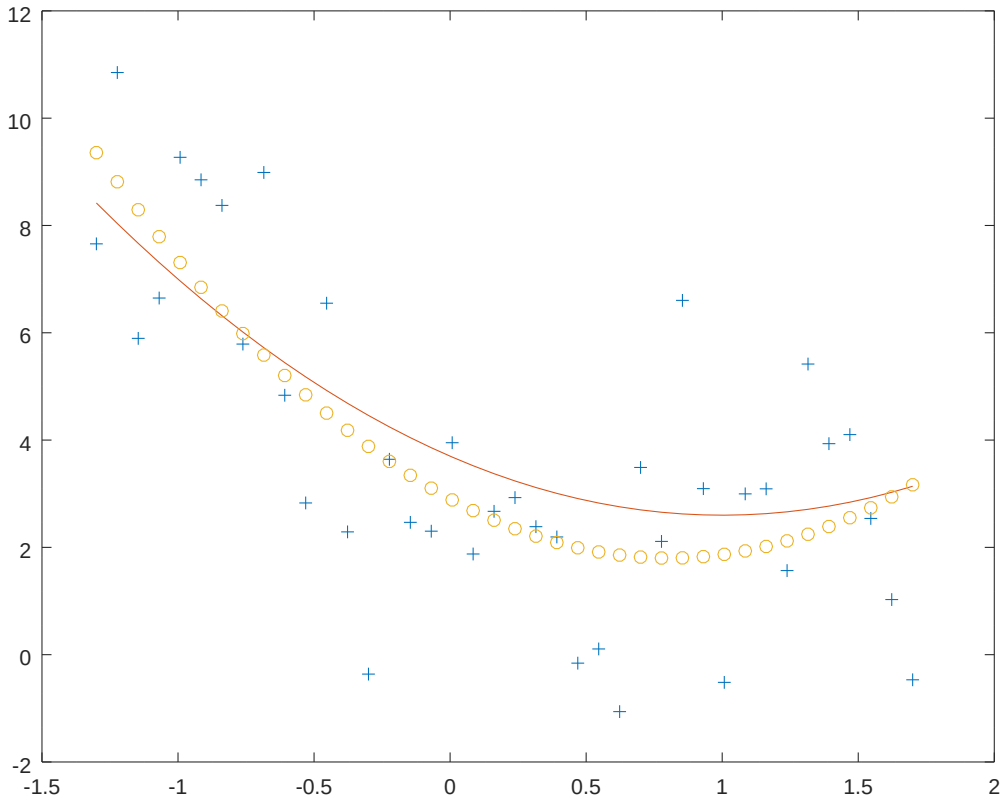
3.1 Код программы

```
pkg load statistics;
clear;
clc;

xmin = -1.3;
xmax = 1.7;
n = 40;
a = [1.1, -2.2, 3.7];
s = 2.7;

X = xmin : (xmax - xmin) / (n - 1) : xmax;
y = polyval(a, X);
Y = y + s * randn(1, n);
polynom = polyfit(X, Y, 2);
Yp = polyval(polynom, X);
plot(X, Y, '+', X, y, X, Yp, 'o');
e = Yp - Y;
printf("Orthogonality check: %f\n", Yp * e');
printf("Noise assessment: %f\n", sqrt(e / (n - 3) * e'));
```

3.2 Вывод программы

```
Orthogonality check: -0.000000
Noise assessment: 2.722484
```

3.3 График



3.4 Вывод

Полученная функция почти совпадает с исходной, вектор невязок ортогонален вектору зашумленной функции, полученный уровень шумов близок к изначальному.

4 Линейная функция

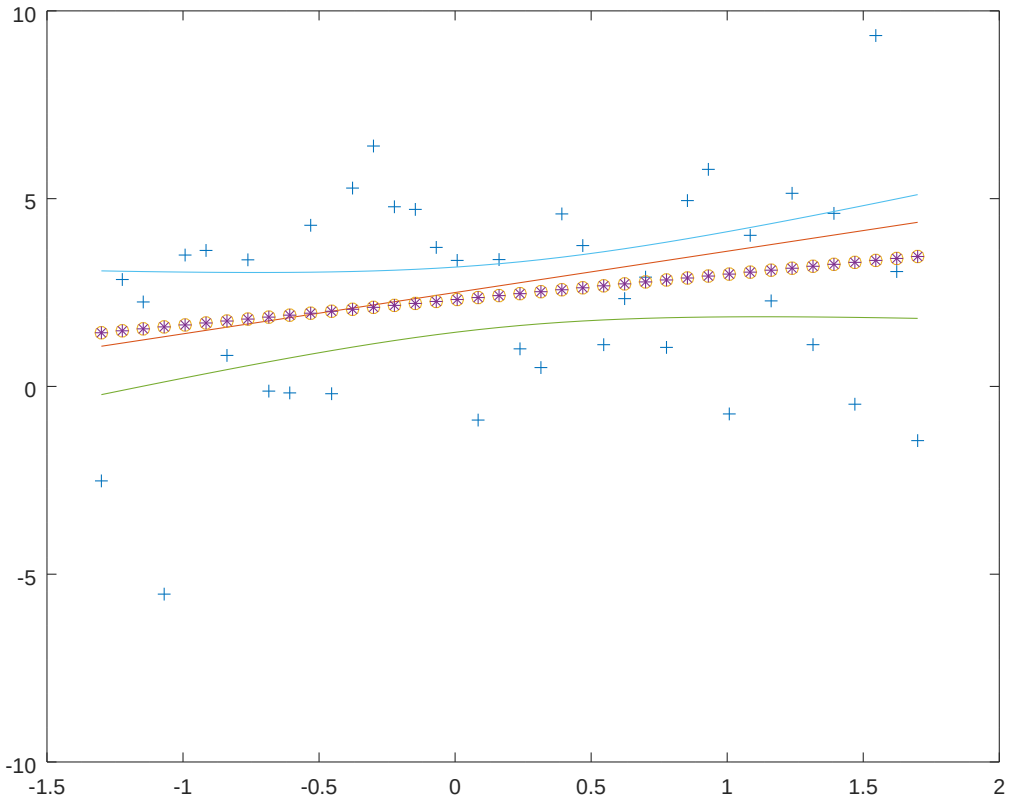
4.1 Код программы

```
pkg load statistics;
clear;
clc;

xmin = -1.3;
xmax = 1.7;
n = 40;
c = [1.1, 2.5];
s = 2.7;

X = xmin : (xmax - xmin) / (n - 1) : xmax;
y = polyval(c, X);
Y = y + s * randn(1, n);

xn = mean(X);
yn = mean(Y);
cov = (X - xn) * (Y - yn)' / (n - 1);
b = cov / (std(X)^2);
Yp1 = yn + b * (X - xn);

polynom = polyfit(X, Y, 1);
printf("Check coef. diff: %f\n", polynom(1) - b);
Yp2 = polyval(polynom, X);

e = Yp1 - Y;
printf("Orthogonality check: %f\n", Yp1 * e');
sn = sqrt(e / (n - 2) * e');
printf("Noise assessment: %f\n", sn);

t = 1.96;
h = t * (sn / sqrt(n));
d = h * (1 + (X - xn).^2 / (std(X)^2)).^(1 / 2);
Yl = Yp1 - d;
Yr = Yp1 + d;

plot(X, Y, '+', X, y, X, Yp1, 'o', X, Yp2, '*', X, Yl, X, Yr);
```

4.2 Вывод программы

```
Check coef. diff: 0.000000
Orthogonality check: -0.000000
Noise assessment: 2.718014
```

4.3 График



4.4 Вывод

Две формулы восстановления ведут себя одним и тем же образом, полученный коэф. равен изначальному, график лежит полностью в доверительном интервале, два вектора ортогональны, полученный уровень шумов близок к изначальному.