МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра Вычислительной техники

практическая работа №3

по дисциплине: Математические методы распознавания образов

«Оценка параметров линии с помощью МНК»

Новосибирск, 2023

ЦЕЛЬ

Получить навыки оценки параметров линии с помощью различных разновидностей метода наименьших квадратов.

ЗАДАНИЕ

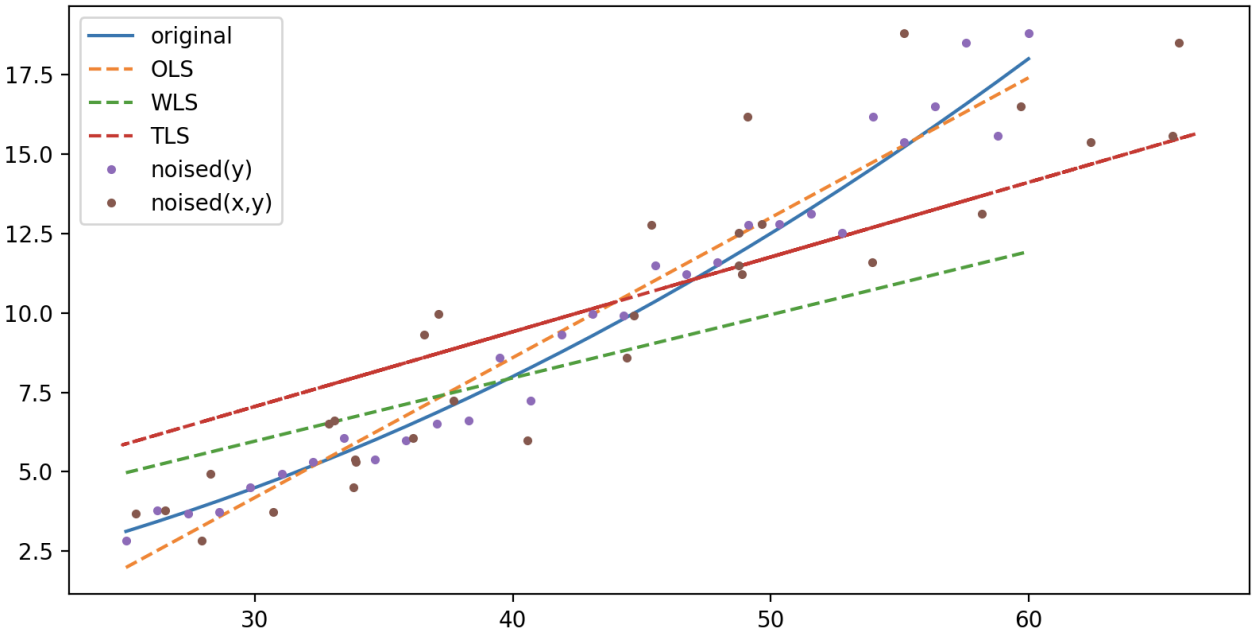
1) Сформировать координаты для несколько линий (3-4)
2) Внести в них шум (10-15%) по кординатам *x* и *y для полного МНК, только по y для остальных.*
3) Определить оценки координат методами:
- Матричного МНК
- Полного МНК
- Взвешенного МНК
4) Выполнить сравнение результатов оценки параметров прямой, полученных разными методами.

ХОД РАБОТЫ

В ходе работы будет использоваться 3 различных функции:

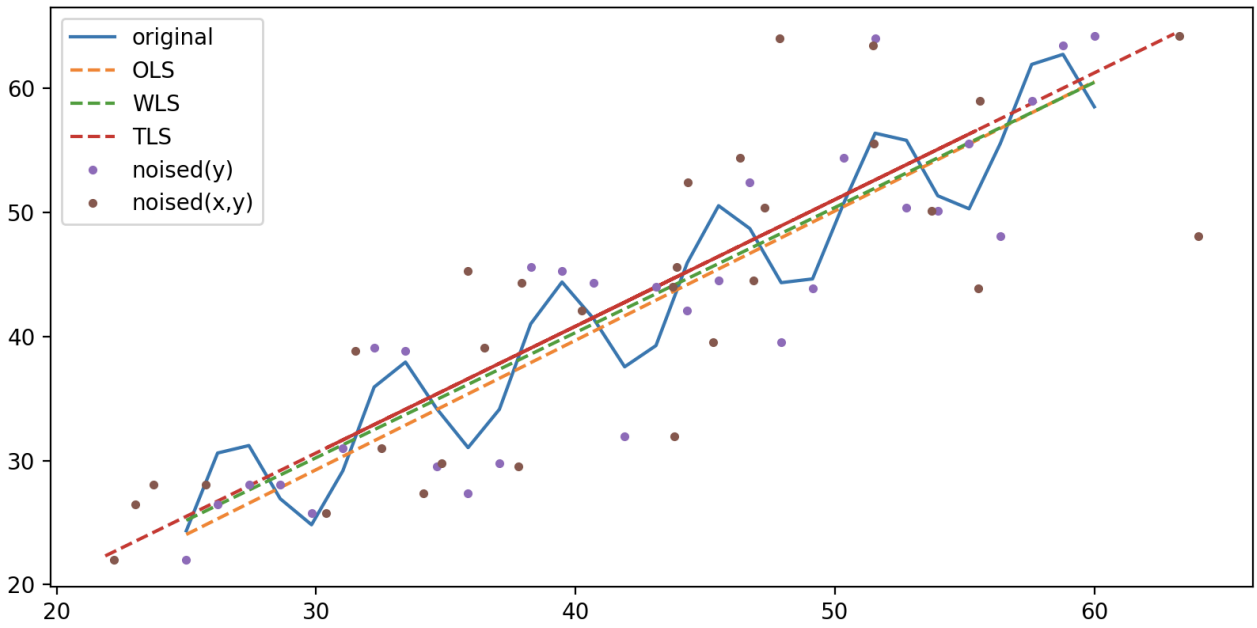
```
def f1(x, *args) -> float:
    return 5 * np.sin(x) + 20 * np.sin(x / 10)


def f2(x, *args) -> float:
    return x + 5 * np.sin(x)

1 usage
def f3(x, *args) -> float:
    return 0.005 * x ^ 2
```

Для каждой из функций программа генерирует 30 точек на интервале от 25 до 60 и накладывает на координату Y шум 15%. Дополнительно для метода TLS (полный МНК) шум накладывается и на координату X.



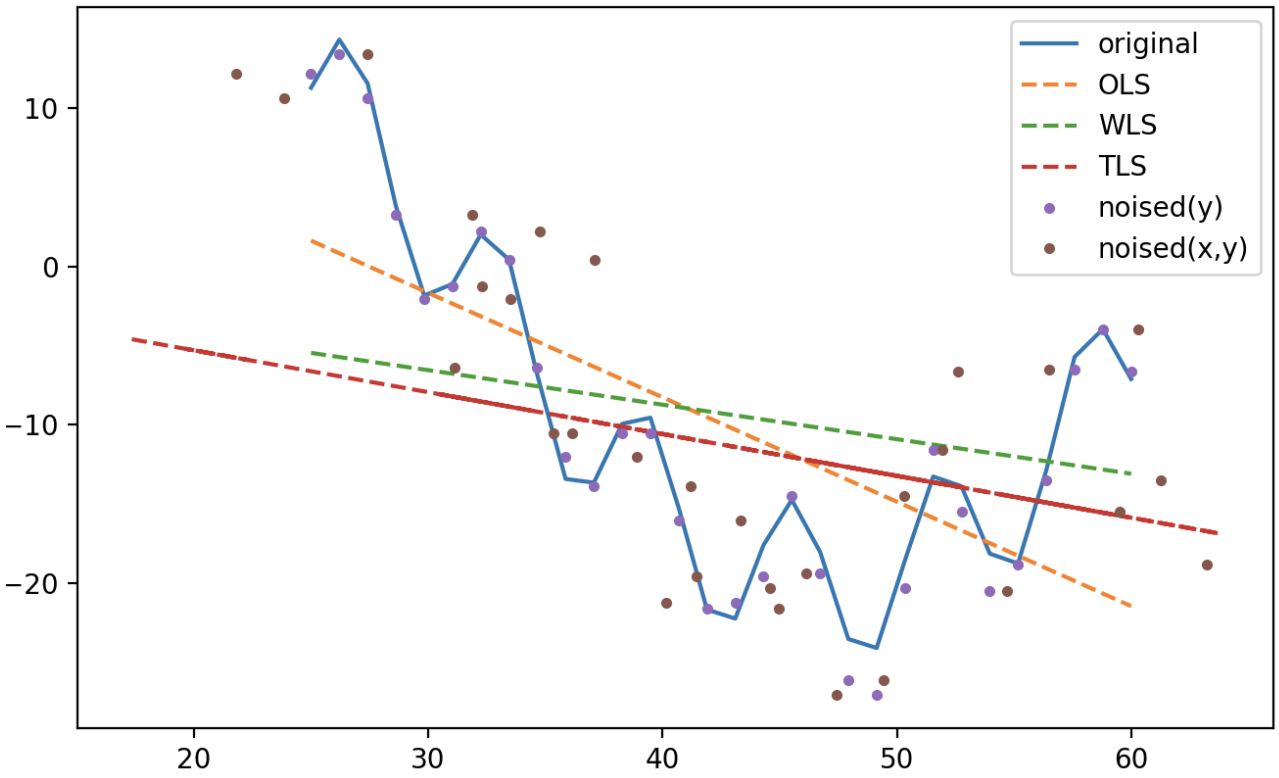
На скриншоте выше показан результат работы программы на функции *f3*. Ниже представлена сводка по ошибке различных алгоритмов:

```
========================================
==============ERROR NORM==============
----------------------------------------
        err_noised: 4.29714
        err_ols: 2.88359
        err_wls: 14.57922
        err_tls: 159.87334
        err_ols_noised: 4.95444
        err_wls_noised: 16.18435
        err_tls_noised: 165.1796
----------------------------------------
```



На скриншоте выше показан результат работы программы на функции *f2*. Ниже представлена сводка по ошибке различных алгоритмов:

```
========================================
==============ERROR NORM==============
----------------------------------------
       err_noised: 22.40613
       err_ols: 19.30337
       err_wls: 19.01173
       err_tls: 469.65704
       err_ols_noised: 29.71809
       err_wls_noised: 29.92847
       err_tls_noised: 495.19605
----------------------------------------
```



На скриншоте выше показан результат работы программы на функции *f3*. Ниже представлена сводка по ошибке различных алгоритмов:

```
========================================
==============ERROR NORM==============
----------------------------------------
       err_noised: 6.68187
       err_ols: 44.38466
       err_wls: 50.31854
       err_tls: 333.47437
       err_ols_noised: 45.8278
       err_wls_noised: 52.45566
       err_tls_noised: 342.04884
----------------------------------------
```

ЗАКЛЮЧЕНИЕ

В ходе работы были получены навыки оценки параметров линии с помощью различных разновидностей метода наименьших квадратов.

Для каждой из трех функций было сгенерировано по 30 точек и были построены графики линейной регрессии, вычисленной с помощью трех алгоритмов: OLS(матричный МНК), TLS(полный МНК) и WLS(взвешенный МНК). Для разных функций результаты получились тоже разные, однако можно сказать, что метод OLS является одним из наиболее точных, т.к. на приведенных выше примерах он показал наименьшую ошибку, что, вероятно, связано с особенностями выбранных для практической работы функций.

ПРИЛОЖЕНИЕ А

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import numpy.linalg as la
import statsmodels.api as sm


mpl.use("MacOSX")


class Noise:
    @staticmethod
    def make_noise(y, p: float, law="uniform"):
        """
        p - уровень шума, от 0 до 1
        """
        eps = abs(y * p)
        if law == "uniform":
            return np.random.uniform(y - eps, y + eps)
        elif law == "normal":
            return np.random.normal(y, eps / 3)



def f1(x, *args) -> float:
    return 5 * np.sin(x) + 20 * np.sin(x / 10)


def f2(x, *args) -> float:
    return x + 5 * np.sin(x)

def f3(x, *args) -> float:
    return 0.005 * x ** 2


def tls(x, y):
    if x.ndim is 1:
        n = 1  # the number of variable of x
        x = x.reshape(len(x), 1)
    else:
        n = np.array(x).shape[1]

    Z = np.vstack((x.T, y)).T
    U, s, Vt = la.svd(Z, full_matrices=True)

    V = Vt.T
    Vxy = V[:n, n:]
    Vyy = V[n:, n:]
    a_tls = - Vxy / Vyy  # total least squares soln
```

```
    xtyt = - Z.dot(V[:, n:]).dot(V[:, n:].T)
    xt = xtyt[:, :n]  # x error
    y_tls = (x + xt).dot(a_tls)
    fro_norm = la.norm(xtyt, 'fro')

    return y_tls, x + xt, a_tls, fro_norm


func = f1

p = 0.15  # процент шума, от 0 до 1
N = 30  # количество точек
x_min, x_max = 25, 60

X = np.linspace(x_min, x_max, N)
y = func(X)
print(X)
print(y)

# noising
x_noised = Noise.make_noise(X, p)
y_noised = Noise.make_noise(y, p)

print(x_noised)
print(y_noised)

############################
# OLS
fit_ols = sm.OLS(y_noised, sm.add_constant(X)).fit()

print(fit_ols.summary())
print("Parameters: ", fit_ols.params)
print("Standard errors: ", fit_ols.bse)
print("R2: ", fit_ols.rsquared)

############################
# WLS

weights = np.ones(N)
weights[N * 6 // 10:] = 3
weights = 1.0 / (weights ** 2)
fit_wls = sm.WLS(y, X, weights=weights).fit()

print(fit_wls.summary())
print("Parameters: ", fit_wls.params)
print("Standard errors: ", fit_wls.bse)
print("R2: ", fit_wls.rsquared)

############################
# TLS

y_tls, x_tls, a_tls, from_norm = tls(x_noised, y_noised)
```

```
##########################
# PREPARE

y_ols = fit_ols.fittedvalues
y_wls = fit_wls.fittedvalues

##########################
# STATS


print("====================================")
print("==============ERROR NORM=============")
print("------------------------------------")
print(f'      err_noised: {round(np.linalg.norm(y - y_noised), 5)}')
print(f'      err_ols: {round(np.linalg.norm(y - y_ols), 5)}')
print(f'      err_wls: {round(np.linalg.norm(y - y_wls), 5)}')
print(f'      err_tls: {round(np.linalg.norm(y - y_tls), 5)}')
print(f'      err_ols_noised: {round(np.linalg.norm(y_noised - y_ols), 5)}')
print(f'      err_wls_noised: {round(np.linalg.norm(y_noised - y_wls), 5)}')
print(f'      err_tls_noised: {round(np.linalg.norm(y_noised - y_tls), 5)}')
print("------------------------------------")

##########################
# plotting

plt.plot(X, y, "-", label="original")
plt.plot(X, fit_ols.fittedvalues, "--", label="OLS")
plt.plot(X, fit_wls.fittedvalues, "--", label="WLS")
plt.plot(x_tls, y_tls, "--", label="TLS")
plt.plot(X, y_noised, "o", label="noised(y)", markersize=3)
plt.plot(x_noised, y_noised, "o", label="noised(x,y)", markersize=3)
plt.legend()
plt.show()
```