Теория Фильтрации и Прогнозирование данных

Липатов Данила МСМТ 243

Лабораторная работа №2

Пункт 1

График полинома второго порядка для X.
Нахождение полинома производилось через встроенные функции ЯП Python: numpy

np.polyval(coef, time)
np.polyfit(time, signal, deg)

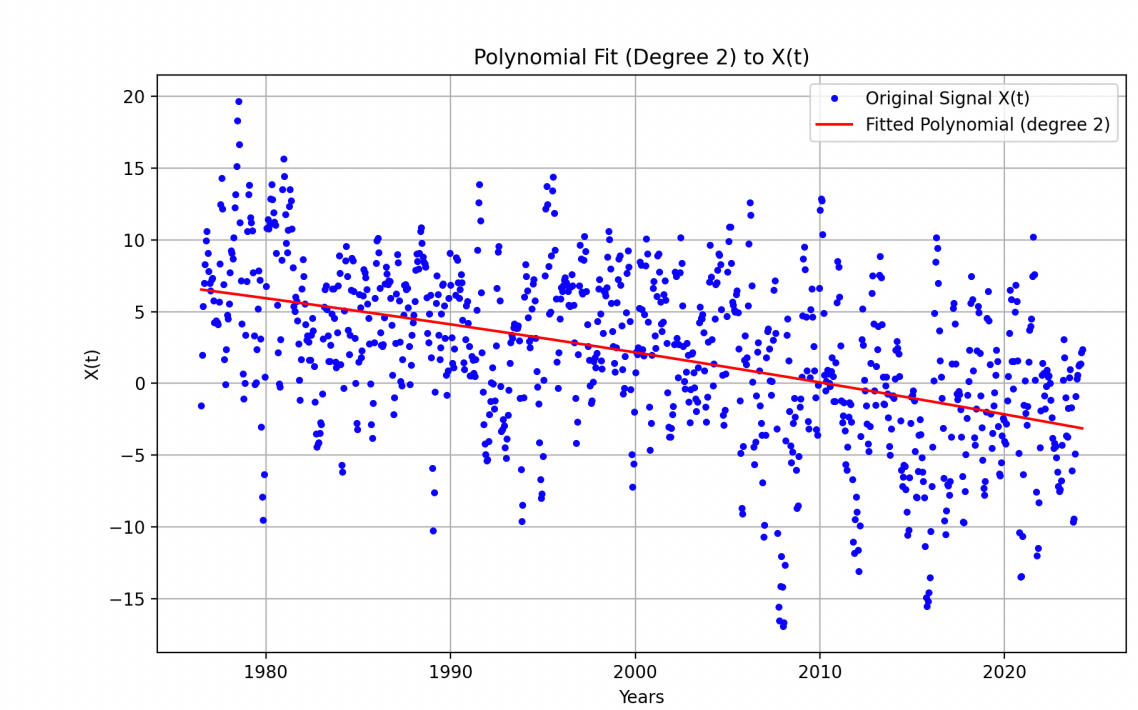

Рис. 1 График полинома 2 рода для X.

Пункт 2

Для нахождения АКФ сигнала X воспользуемся встроенной функцией numpy.correlate(), в качестве аргументов передается сигнал.

```
def autocovariance(signal_):
    N = len(signal_)
    acf = np.correlate(signal_, signal_, mode='full')/ N
    lags = signal.correlation_lags(len(signal_), len(signal_))
    return acf, lags
```

Для каждого сигнала (a, b) найдем его АКФ

```
X_mean = X - np.mean(X)
X_detrended = X - polyn
acf_mean, lags_mean = autocovariance(X_mean)
acf_detrended, lags_detrended = autocovariance(X_detrended)
```

Получим следующий графики АКФ для каждого из варинатов:

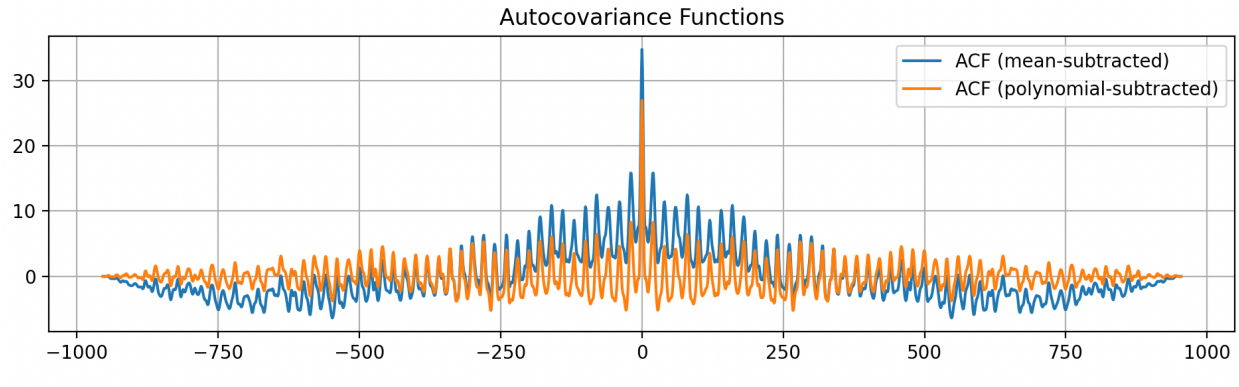

Рис. 2 АКФ для X_mean / X_detrended

Аналогично алгоритмам, реализованных в лаб.работе №1 БФП найдем спектры АКФ для обеих пунктов и отдельно для сигнала X, получим следующее:

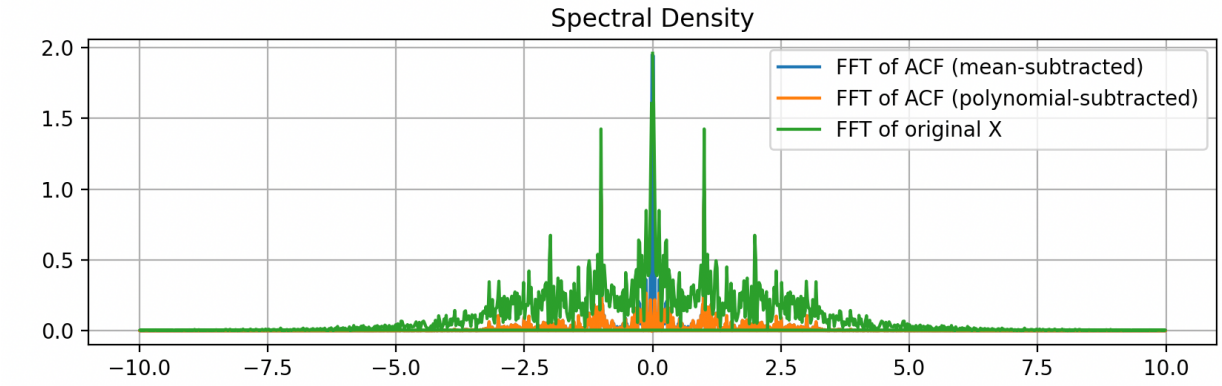

Рис. 3 Спектрограммы для АКФ и X

Пункт 3

Для вычисления кросс-корреляционную функцию достаточно применить функцию из пункта 2, где вторым аргументом будет сигнал Y.
Получим следующий график:

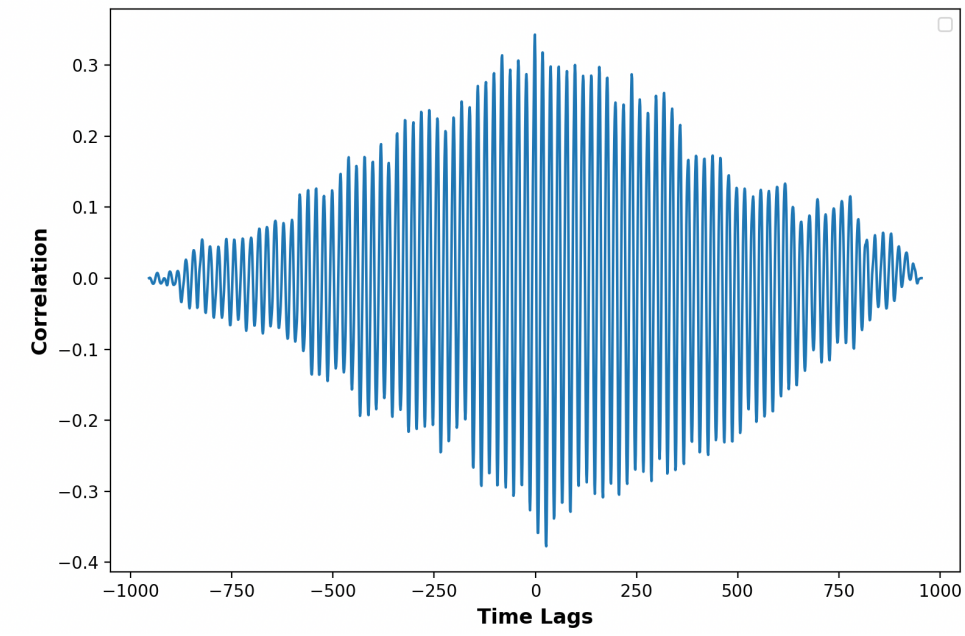

Рис. 4 Кросс-корреляционая функция между X и Y

Кросс-спектр между X и Y

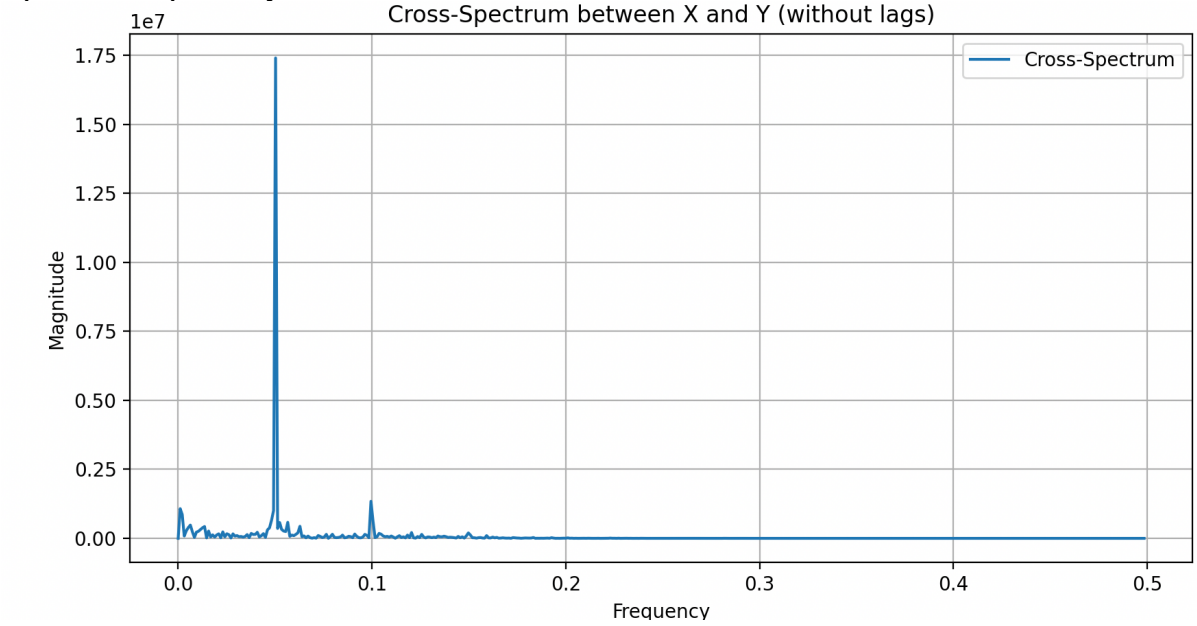

Рис. 5 Кросс-спектр между X и Y

Пункт 4

Для генерации случайный сигнал АРСС (ARMA) используем следующие коэффициенты и функции в Python

```
ar = np.array([0.5, -0.25])  # Коэффициенты AR (авторегрессия)
ma = np.array([0.5, -0.3])  # Коэффициенты MA (скользящее среднее)
arma_process = ArmaProcess(ar, ma)
n_samples = 1000
ARMA = arma_process.generate_sample(nsample=n_samples)
```

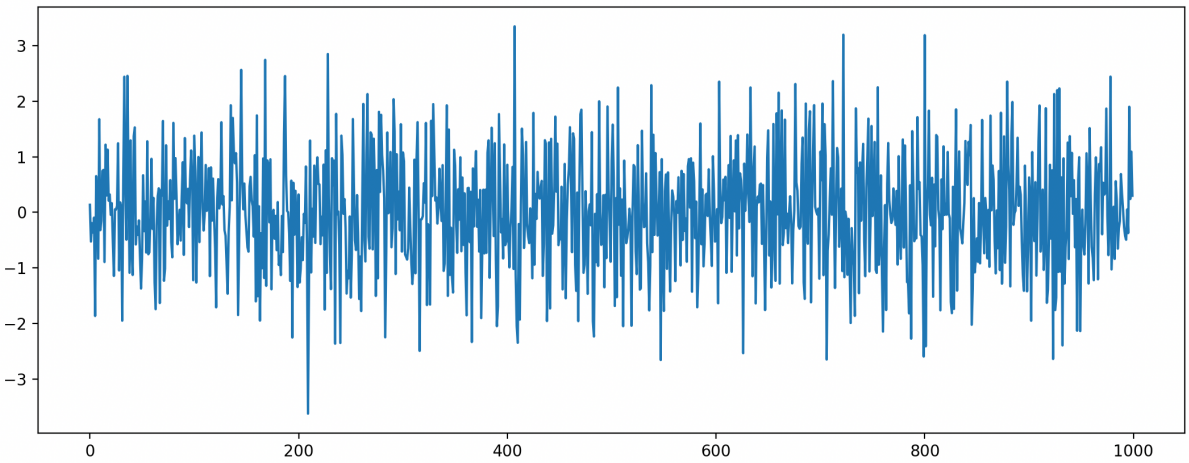

Рис. 6 Смоделированный сигнал ARMA

Так же необходимо было построить смещенную и несмещенную АКФ ARMA.
Для этого воспользуемся так же функцией np.correlate(), предварительно рассчитывая необходимую АКФ

if biased:
result /= n # Смещённая АКФ
else:
result = result[n - 1:]
result /= (n - np.arange(n)) # Несмещённая АКФ

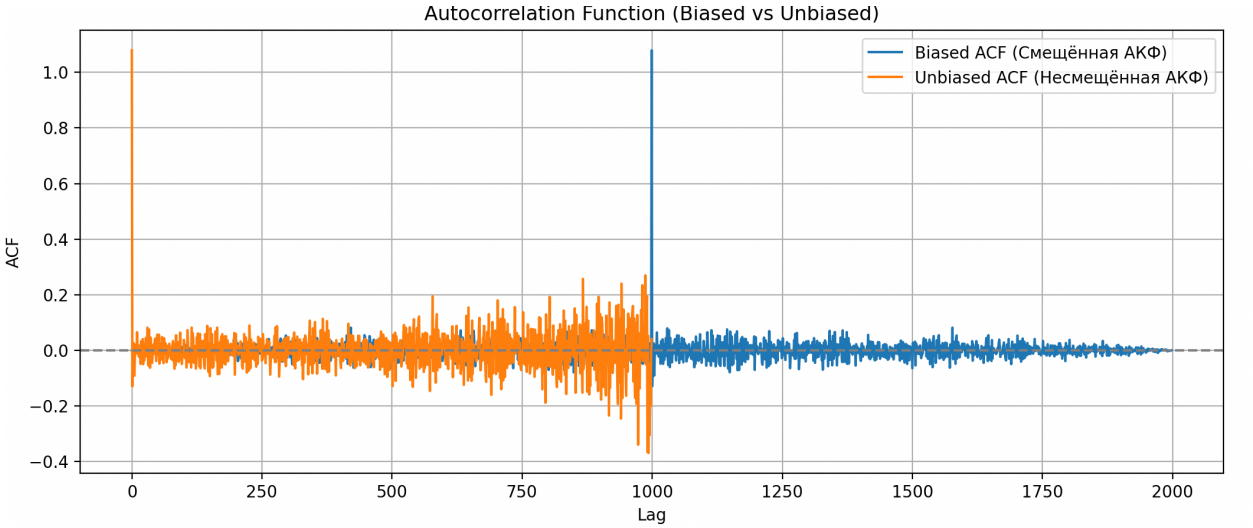

Рис. 7 Графики смещенной и несмещенной АКФ (ограниченные слева и размещенные, чтобы проще было понять поведение)

Примечание:
Несмещённая АКФ компенсирует уменьшение количества точек при больших лагах, что приводит к большим колебаниям её значений на концах ряда
Для оценки устойчивости корней в Python у функции ArmaProcess(ar, ma) есть метод проверки на устойчивость для ar и ma отдельно

```
>>> arma_process.isstationary
True
>>> arma_process.isinvertible
True
```