

Теория Фильтрации и Прогнозирование данных

Липатов Данила МСМТ 243
Лабораторная работа №2

Пункт 1

Полином второго порядка имеет вид:

$$P(x) = a_0 + a_1x + a_2x^2$$

График полинома второго порядка для X.

Нахождение полинома производилось через встроенные функции ЯП Python: numpy

np.polyval(coef, time)

np.polyfit(time, signal, deg)

Коэффициенты полинома

-6.76844480e-04 2.50515932e+00 -2.30078157e+03

Где

-6.76844480e-04 - маленькое значение говорит о слабом квадратичном тренде, так как он отрицательный, ветви параболы направлены вниз

2.50515932 - отвечает за линейный тренд, показывая увеличение на каждом временном шаге, чем больше значение, тем больше наклон графика.

-2300.7 – Пересечение при $x = 0$

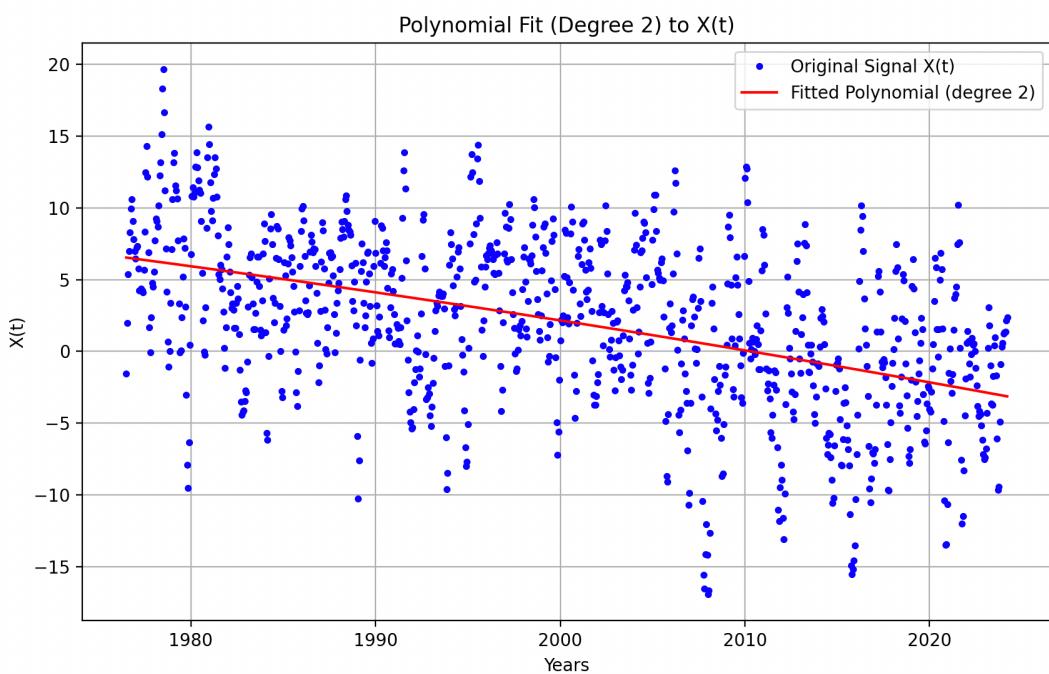


Рис. 1 График полинома 2 рода для X.

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 вариантов:

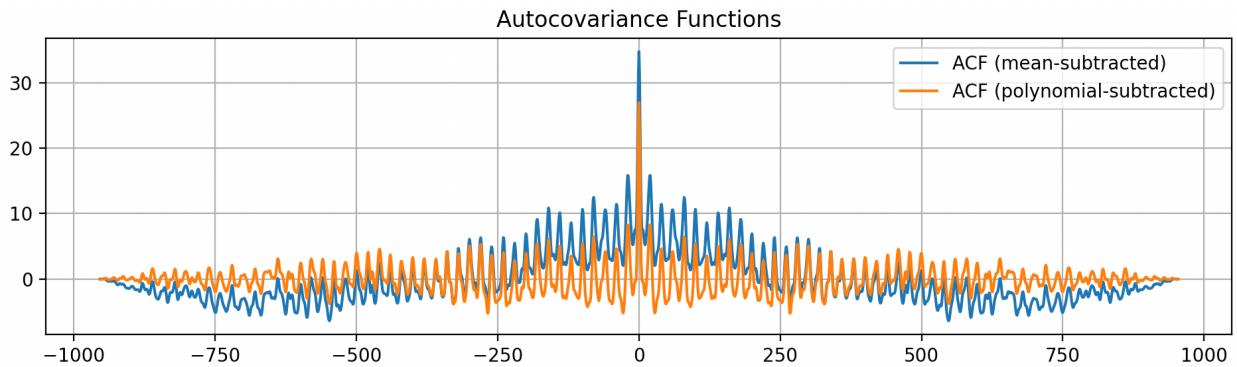


Рис. 2 АКФ для $X_{\text{mean}} / X_{\text{detrended}}$

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

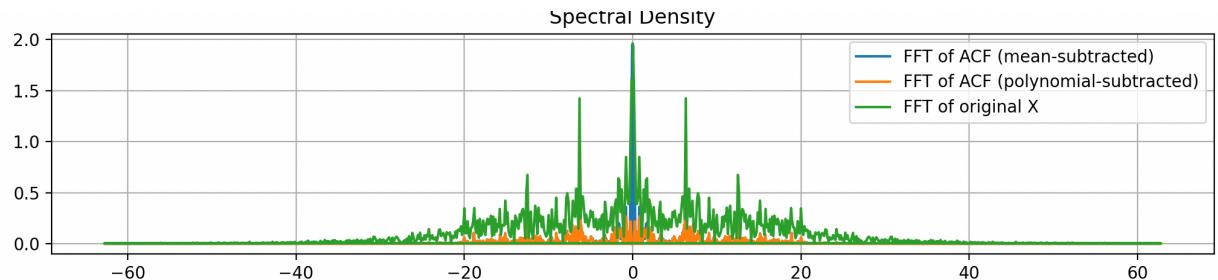


Рис. 3 Спектрограммы для АКФ и X
Spectral Density

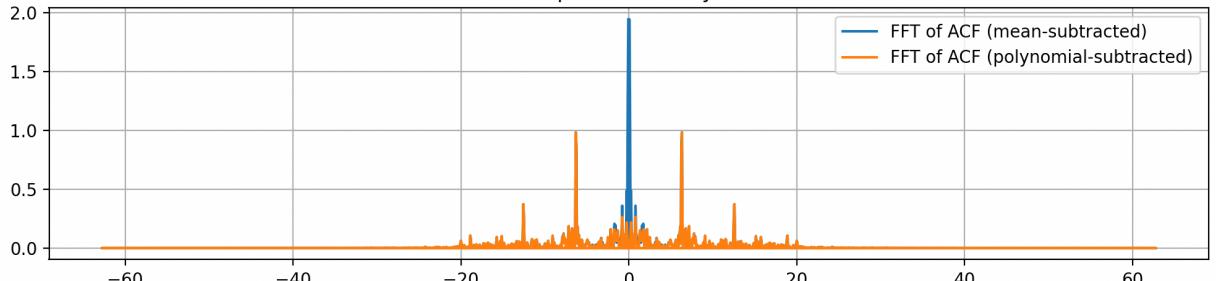


Рис. 3.2 Спектрограммы для АКФ и X

При увеличении спектрограммы можно заметить, что АКФ с вычетом полинома в 0 имеет 0-ую амплитуду. А если мы посмотрим наложение одного спектра на другой, то в целом можно заметить схожее поведение со спектром оригинального сигнала.

Пункт 3

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

Получим следующий график:

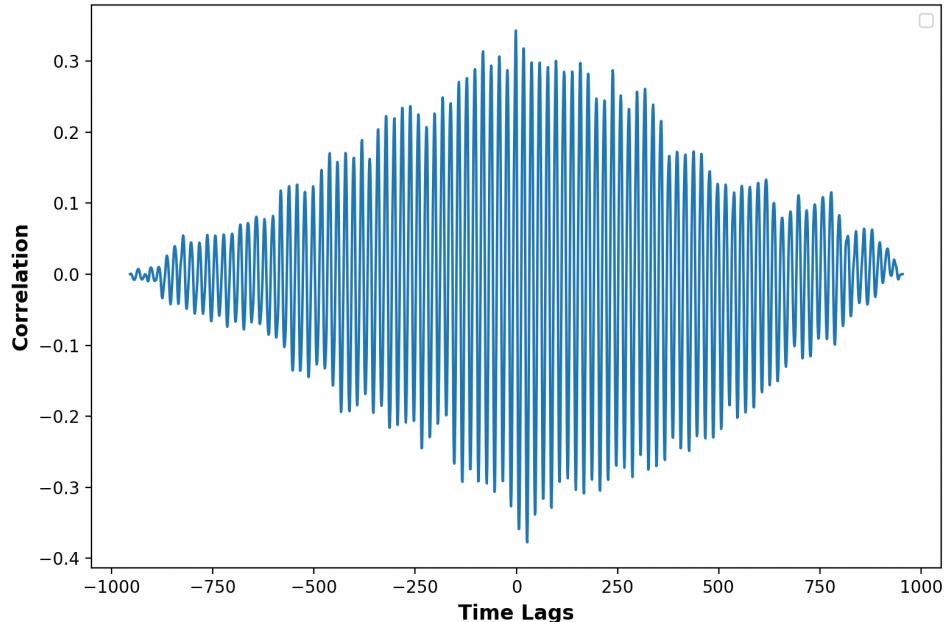


Рис. 4 Кросс-корреляционная функция между X и Y

В целом из графика видно, что в окрестности 0 находится максимальное значение. Однако найдем точное значение лага, которому соответствует максимальное значение. Это означает, что второй временной ряд опережает первый на один временной шаг.

Кросс-спектр между X и Y

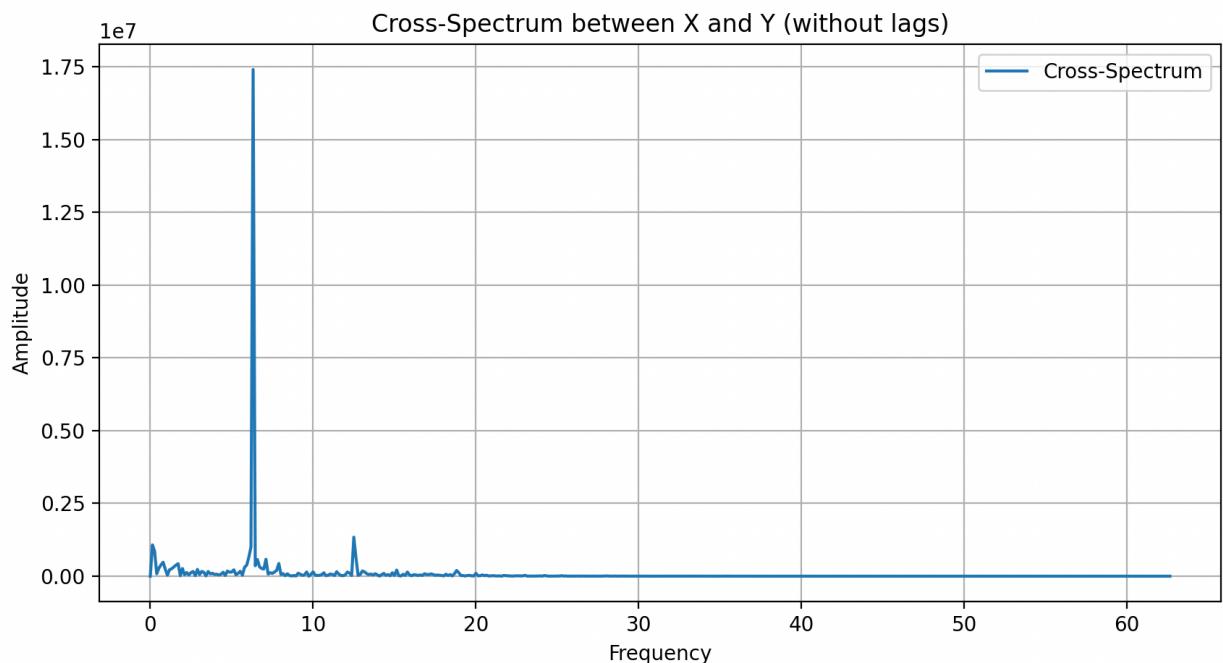


Рис. 5 Кросс-спектр между X и Y

Пункт 4

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

```
ar_1 = 0.5
ar_2 = 0.25
ar_3 = 0.1
ma_1 = -0.3
ma_2 = -0.2
ma_3 = 0.1

ar_root = np.array([-ar_1, -ar_2, -ar_3, 1]) # Коэффициенты AR (авторегрессия)
ma_root = np.array([-ma_1, -ma_2, -ma_3, 1]) # Коэффициенты MA (скользящее среднее)
arma_process = ArmaProcess(ar, ma)
n_samples = 1000

ARMA = arma_process.generate_sample(nsample=n_samples)
```

Тогда корни полинома будут следующие (отдельно для AR / MA)

```
[-0.7837762+1.12211372i -0.7837762-1.12211372i 1.0675524+0.i]
[-1.83649528+0.i 0.58491431+1.2136419i 0.58491431-1.2136419i]
```

Взяв модуль каждого корня увидим, что все они лежат вне $|z| > 1$

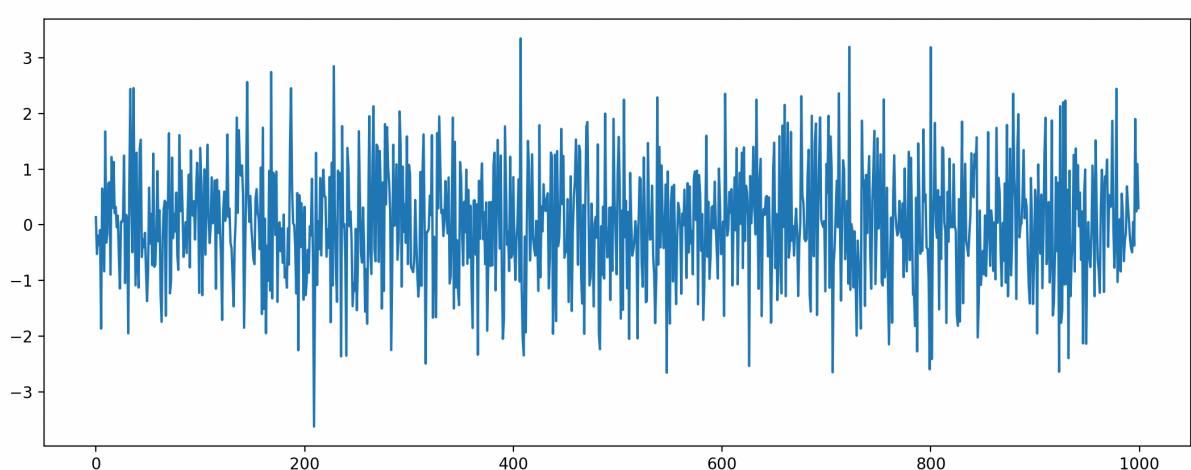


Рис. 6 Смоделированный сигнал ARMA

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 # Смешённая АКФ
else:
    result = result[n - 1:]
result /= (n - np.arange(n)) # Несмешённая АКФ
```

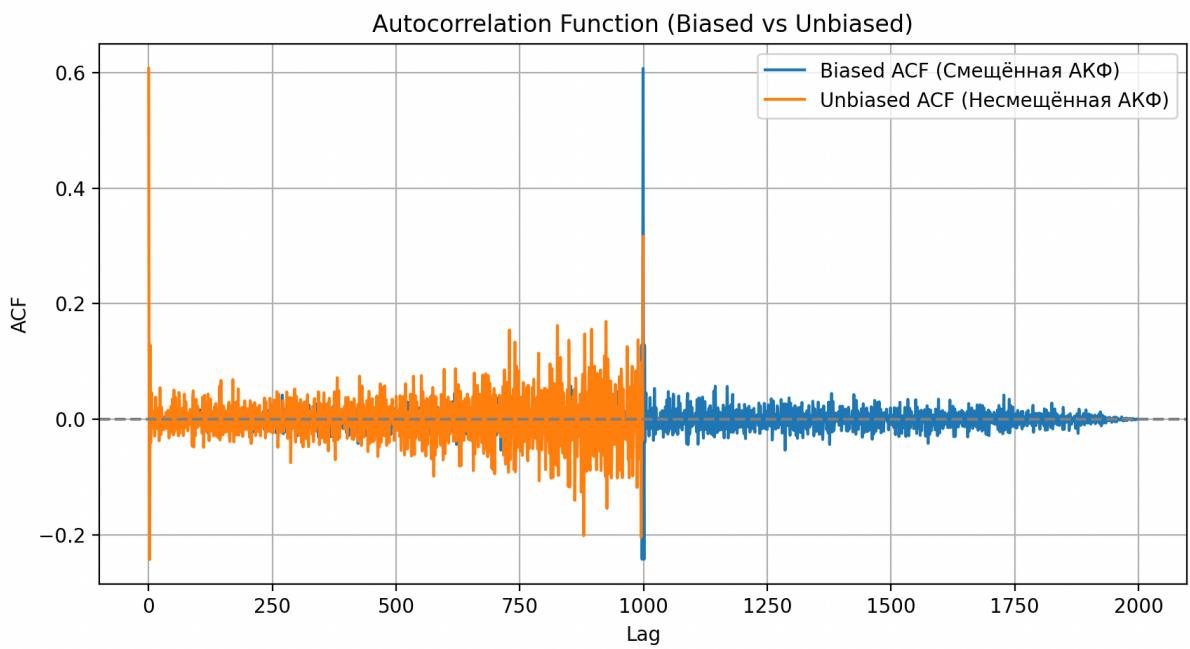


Рис. 7 Графики смещенной и несмещенной АКФ (ограниченные слева и размещенные, чтобы проще было понять поведение)

АКФ смещенная:

$$\frac{1}{N} \sum_{t=1}^{N-k} (x_t - \hat{x})(x_{t+k} - \hat{x})$$

АКФ несмещенная:

$$\frac{1}{N-k} \sum_{t=1}^{N-k} (x_t - \hat{x})(x_{t+k} - \hat{x})$$

Изменяя нормировочный делитель с

N на $N-k$, мы переходим с смещенной на несмещенную оценку

Spectral Density of ARMA Process

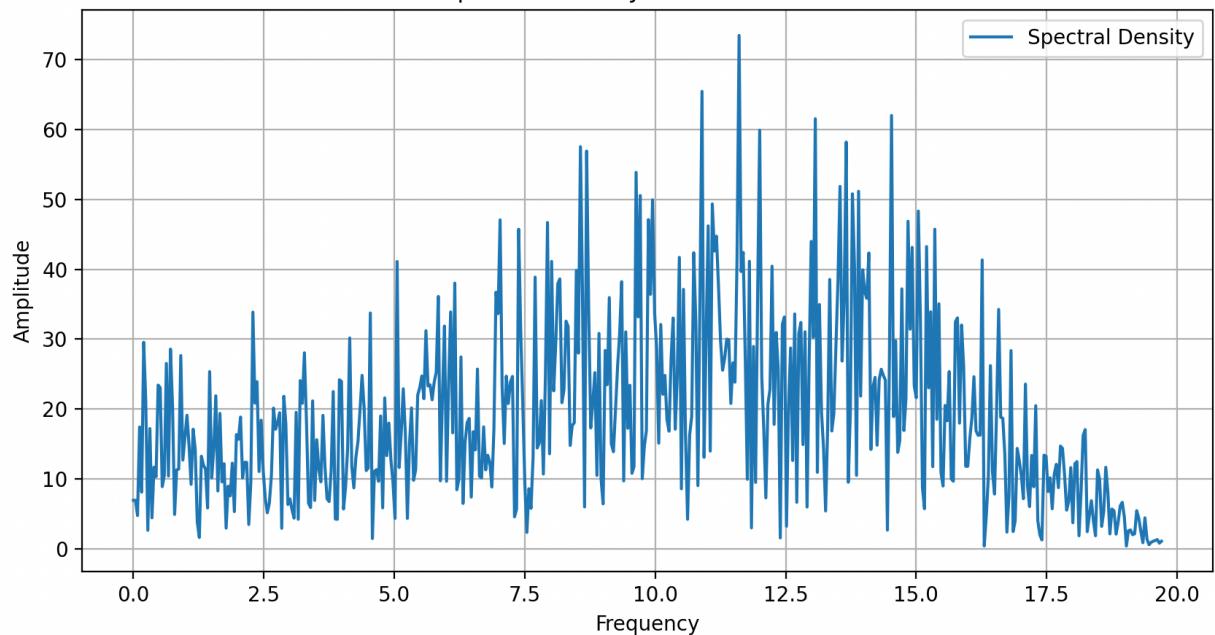


Рис. 8 Спектр произвольной ARMA модели

Примечание:

Несмешённая 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
```