Анализ данных

Хашин С.И.
http://math.ivanovo.ac.ru/dalgebra/Khashin/index.html

Ивановский университет

.

Временные ряды

.

Иваново-2023

План

Данные в папке time_series

1. mpi_roof_2008.npz Метео данные из в Германии за 2008 год с шагом 10 мин. Всего 52704 записи от 01.01.2008 00:10:00 до 01.01.2009 00:00:00.
2. brent.npz ежедневные биржевые цены на нефть brent с 20.05.1987 по 28.08.2020. Вектор длины 12155.
3. wti.npz ежедневные биржевые цены на нефть wti с 02.01.1986 по 31.08.2020. Вектор длины 12661.
4. hotwater.npz часовое потребление горячей воды объектом, 2020 г. 9529 строк по 11 чисел в каждой (с 6 утра).
5. GermanE.npz.npz Суточная выработка ээ в Германии с 01.01.2006 по 31.12.2017, 4384 строки по 3 числа в каждой.
6. Цены на драг.металлы СБ 2000-2016

mpi_roof_2008.npz

Давление и температура на метеостанции в Германии с шагом 10 мин. с 2003 по 2023 года, всего свыше 1 млн.записей.
Исходный файл:

```
"Date Time","p (mbar)","T (degC)"
24.11.2003 16:00:00,  980.85,   13.20
24.11.2003 16:10:00,  980.85,   13.09
24.11.2003 16:20:00,  980.85,   12.92
24.11.2003 16:30:00,  980.85,   12.66
24.11.2003 16:40:00,  980.86,   12.54
24.11.2003 16:50:00,  980.86,   12.44
24.11.2003 17:00:00,  980.85,   12.01
...
07.10.2023 18:50:00,  992.93,   17.15
07.10.2023 19:00:00,  992.88,   17.02
07.10.2023 19:10:00,  992.98,   17.01
```

mpi_roof_2008.npz

№	Name	Описания
0	p (mbar)	Атмосферное давление в mbar
1	T (degC)	Температуа
2	VPmax (mbar)	Давление насыщенного пара
3	VPact (mbar)	Давление пара
4	sh (g/kg)	Удельная влажность
5	wv (m/s)	Скорость ветра
6	max. wv (m/s)	Максимальная скорость ветра
7	wd (deg)	Направление ветра в градусах

Как читать?

```
dataz = np.load("time_series\\mpi_roof_2008.npz")
for name, matrix in dataz.items():
    print(name, "\t", matrix.shape) # data    (52704, 8)
data = dataz['data']
print('shape: ', data.shape)          # shape:  (52704, 8
```

brent.npz wti.npz

Два файла: brent-daily_csv.csv, wti-daily_csv.csv

brent-daily_csv.csv

```
Date,Price
20.05.1987,18.63
21.05.1987,18.45
22.05.1987,18.55
25.05.1987,18.60
26.05.1987,18.63
27.05.1987,18.60
...
25.08.2020,46.01
26.08.2020,45.79
27.08.2020,44.84
28.08.2020,45.22
```

wti-daily_csv.csv

```
Date,Price
02.01.1986,25.56
03.01.1986,26.00
06.01.1986,26.53
07.01.1986,25.85
08.01.1986,25.87
09.01.1986,26.03
...
26.08.2020,43.21
27.08.2020,42.88
28.08.2020,42.96
31.08.2020,42.61
```

brent.npz wti.npz

Как читать?

```
data = np.load("time_series\\brent.npz")['data']
print('shape: ', data.shape) # shape:  (12155,)

data = np.load("time_series\\wti.npz")['data']
print('shape: ', data.shape) # shape:  (12661,)
```

Потребление тепла

Почасовое потребление горячей воды у объекта.
field_dedscription.txt - описание полей
Дополнительно укажу только два поля:
date — время в секундах по Гринвичу.
dateOnEndOfArchive — должно равняться date+3600, время окончания интервала, на 1 час больше.

hotWater.npz

Почасовое потребление горячей воды, обработанный файл.

0	t1	Температура на подающем трубопроводе, °C
1	t2	Температура на обратном трубопроводе, °C
2	V1	Объём на подающем трубопроводе, м3
3	V2	Объём на обратном трубопроводе, м3
4	V3	Объём на подпитке, м3
5	M1	Масса на подающем трубопроводе, т
6	M2	Масса на обратном трубопроводе, т
7	M3	Масса на подпитке, т
8	P1	Давление на подающем трубопроводе, МПа
9	P2	Давление на обратном трубопроводе, МПа
10	Q	Количество тепловой энергии по всей системе, Гкал

Выработка и потребление ээ в Германии с 2003 г.

```
0   Consumption   Всего
1   Wind          Ветер
2   Solar         Солнце
```

```
Как читать?
data = np.load("time_series\\GermanE.npz")['data']
print('hotWater.npz: ', data.shape)
> hotWater.npz:  (4383, 3)
В начале ветра и солнца нет:
1069.184,0,0
1380.521,0,0
1442.533,0,0
...
```

Цена золота по дням

Цены на драг.металлы СБ от 26.04.2000 до 26.10.2016.

```
Date,Gold,Silver,Platinum,Palladium,,руб.г.
26.10.2016,2532.11,35.37,1897.14,1276.73,,
25.10.2016,2535.14,35.30,1872.84,1252.56,,
22.10.2016,2537.77,35.16,1859.23,1248.86,,
21.10.2016,2547.07,35.32,1886.42,1270.32,,
20.10.2016,2554.89,35.59,1899.45,1285.75,,
19.10.2016,2551.00,35.69,1914.80,1298.10,,
...
03.05.2000,254.51,4.59,481.34,533.60,,
28.04.2000,253.09,4.56,467.92,547.74,,
27.04.2000,253.59,4.56,437.19,523.52,,
26.04.2000,255.32,4.56,434.89,528.67,,
```

Цена золота по дням

Цены на драг.металлы СБ от 26.04.2000 до 26.10.2016.
4 столбца: Gold,Silver,Platinum,Palladium
Как читать?

```
data = np.load("time_series\\gold_sb.npz")['data']
print('gold_sb.npz: ', data.shape)
> gold_sb.npz:  (6028, 4)
```

Читаем всё!

```
for fname in ('mpi_roof_2008.npz', 'brent.npz',
              'wti.npz', 'hotWater.npz',
              'GermanE.npz', 'gold_sb.npz'):
    data = np.load("time_series\\"+fname)['data']
    print(fname, ': ', data.shape)
```

```
mpi_roof_2008.npz:(52704, 8)
brent.npz :       (12155,)
wti.npz :         (12661,)
hotWater.npz :    (9529, 11)
GermanE.npz :     (4383, 3)
gold_sb.npz :     (6028, 4)
```

Скользящее окно

```
def sliding_window(X, k): # скользящее окно на X размера [-k
    '''
    скользящее окно на векторе X размера [-k,k]
    :param X: временной ряд
    :param k: размер окна [-k,k]
    :return: сглаженный ряд   '''
    N = len(X)
    Xs = np.zeros(N)          # сглаженный ряд
    for i in range(k):        # слева:
        Xs[i] = np.sum(X[:k+i])/(k+i)
    for i in range(k, N-k):       # в центре:
        Xs[i] = np.sum(X[i-k:i+k+1])/(2*k+1)
    for i in range(k):        # справа:
        Xs[-1-i] = np.sum(X[-i-k:])/(k+i)
    return Xs
```

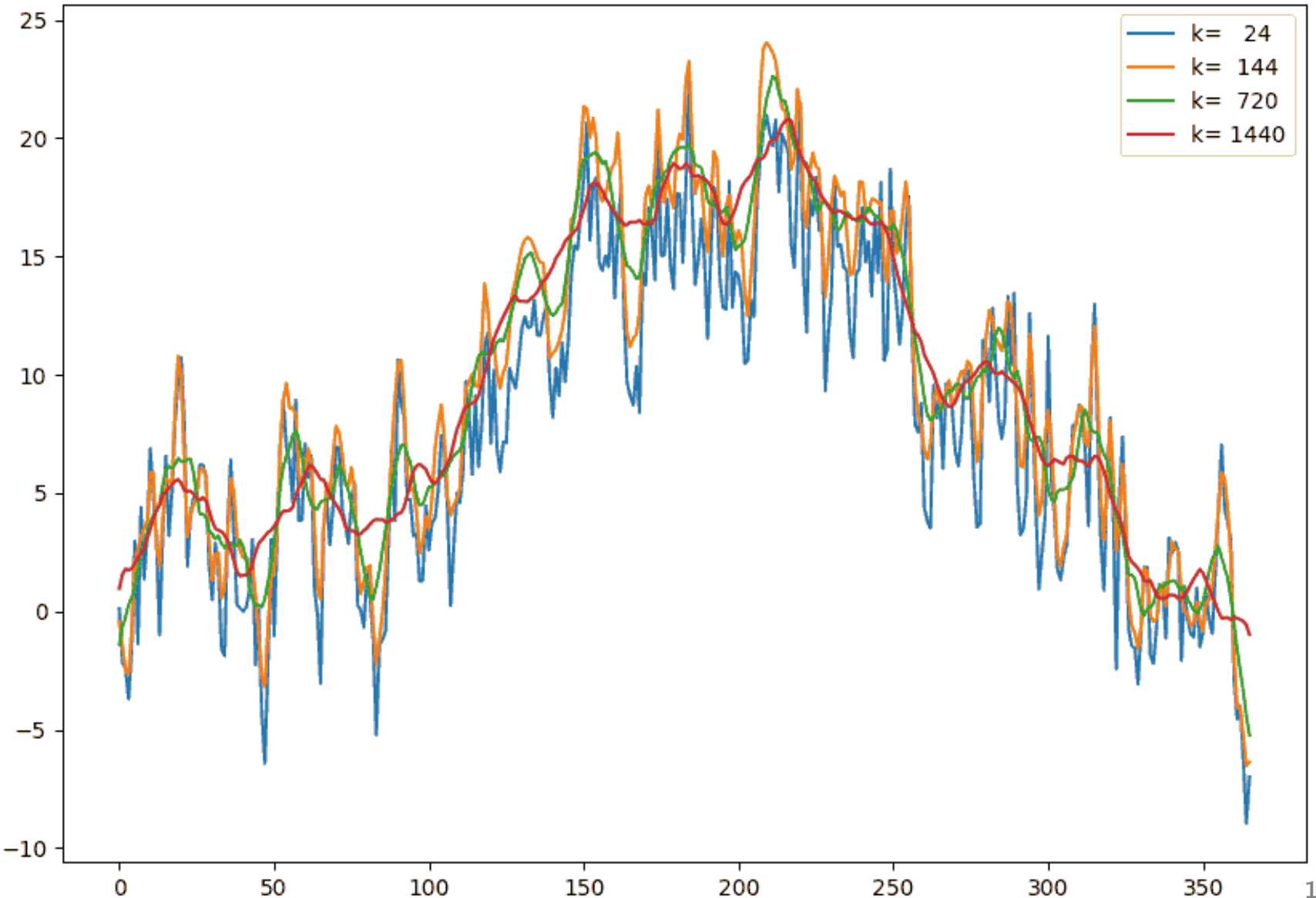
Проверка sliding_window

```
def tst_sliding_window():
    N, k = 10,3
    X = np.ones(N)
    print(sliding_window(X,k))

    > [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

sliding_window

```
A12 = np.loadtxt('t:\\A12.csv', delimiter=',', skiprows=1)
print(A12.shape)
N = len(A12)
for k in (24, 144, 720, 1440):
    cels_s = mlu.sliding_window(A12[:,1], k)
    cy = cels_s[::144]
    plt.plot(np.arange(len(cy)), cy, label=f'k={k:5d}')
    #tk_show(cy, np.arange(len(cy)), 'red', 600, 800)

plt.legend() plt.show()
```

Сглаживание



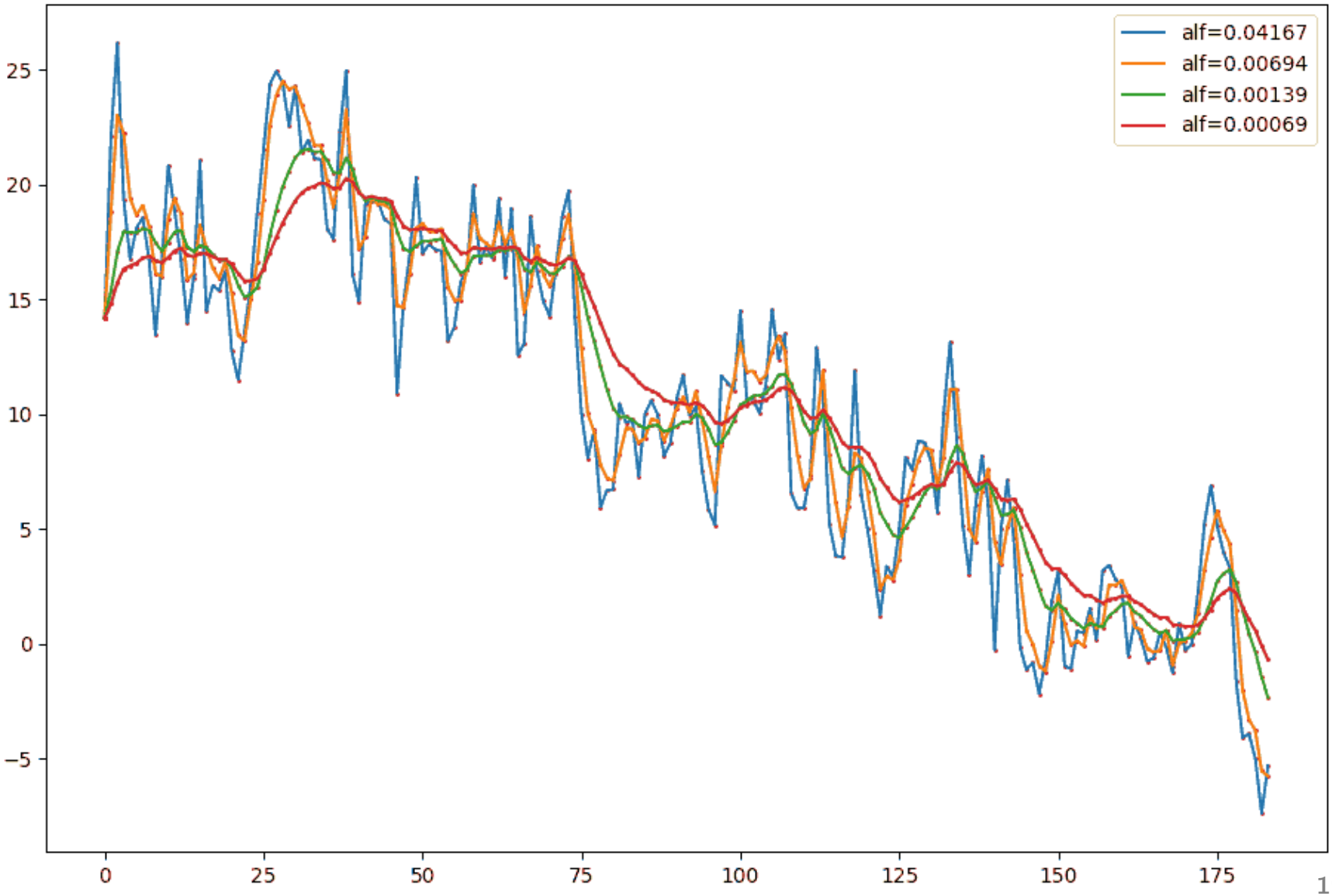
Экспоненциальное сглаживание

Пусть x_i — временной ряд. Сглаженный ряд s_i строим так. Положим $s_0 = \alpha x_0$ и далее рекуррентно находим:

$$s_i = \alpha x_i + (1 - \alpha) s_{i-1}.$$

```
def exp_smoothing(x, alpha):
    # Экспоненцильное сглаживание вектора x
    # с параметром alpha
    x_smoth = np.zeros(len(x))
    x_smoth[0] = x[0]
    for i in range(1, len(x)):
        x_smoth[i] = alpha*x[i] + (1-alpha)*x_smoth[i-1]
    return x_smoth
```

Экспоненциальное сглаживание



Ковариационная функция

Пусть x_i — временной ряд. Его среднее значение:

$$M(x) = \sum x_i / N.$$

и дисперсия σ^2:

$$\sigma^2 = M((x - M(x))^2) = s_2/N - (s_1/N)^2,$$

где

$$s_1 = \sum x_i, \qquad s_2 = \sum x_i^2.$$

Ковариационная функция

Пусть x_i, y_i — два временных рядя. Их ковариация:

$$cov(x, y) = \frac{\sum(x_i - Mx)(y_i - My)}{N\sigma(x)\sigma(y)}$$

Если ряды жёстко зависят друг от друга, то есть колеблются синхронно относительно средних значений, то ковариация близка к 1.
Если же ряды колеблются «в противофазе», то ковариация близка к -1.
Если случайные величины не зависят друг от друга, то ковариация близка к 0.

Ковариационная функция

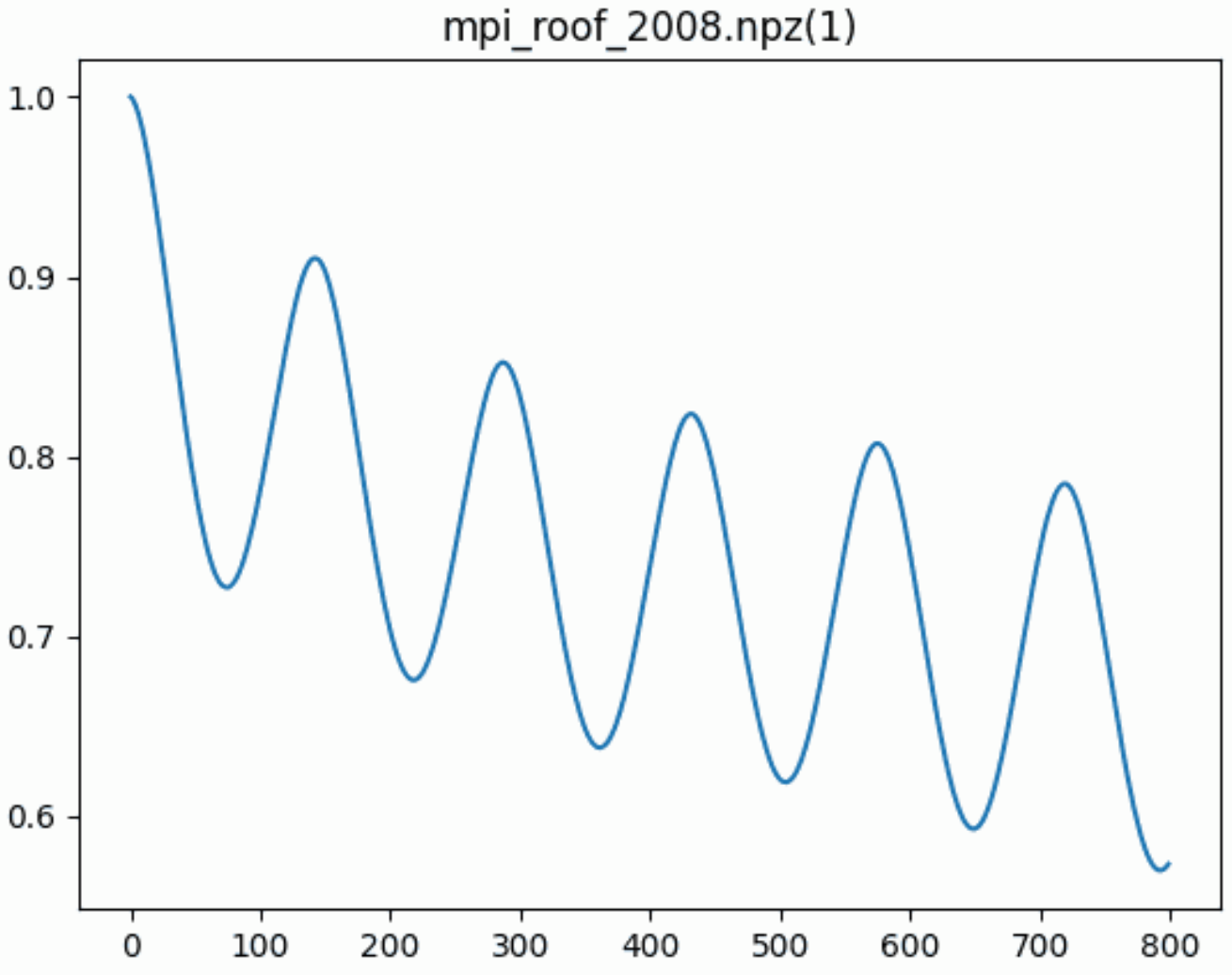
```
def covar(x,y): # ковариация (x,y)
    N = len(x)
    Mx, My = np.mean(x), np.mean(y)
    sc_prod = np.mean((x-Mx)*(y-My))
    denom = np.std(x)*np.std(y)
    return sc_prod/denom

def tst_covar():
    x = np.arange(10)
    print( 'cov=', covar(x,3*x-2), ', must be 1.0')
    print( 'cov=', covar(x,-3*x+7), ', must be -1.0')

tst_covar()
```

Ковариационная функция

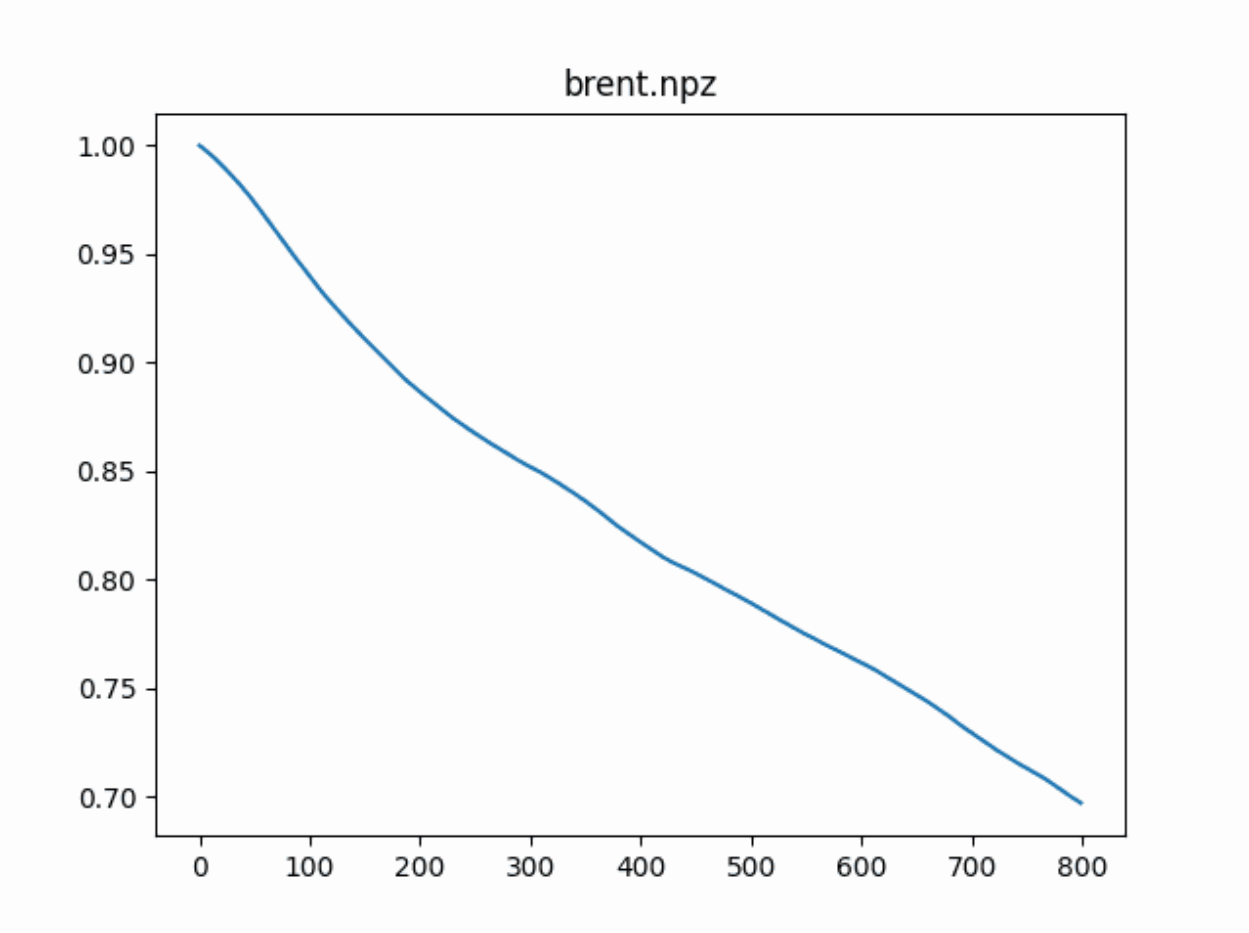
```
def covariation(x, k):
    # ковариационная функция вектора x длины k
    res = np.ones(k)
    for i in range(1,k):
        res[i] = covar(x[i:], x[:-i])
    return res

def tst_covariation():
    N, k = 400, 100
    x = np.sin(10*np.arange(N)/N)+0.02*np.random.random(N)
    cov = covariation(x, k)
    plt.plot(np.arange(k), cov)
    plt.show()

tst_covariation()
```

Ковариационная функция

Температура



Ковариационная функция, биржевые цены



Задача 1. Построить корреляционную кривую для каждого столбца каждой из полученных матриц.
см. time_series_covar/*.png

Тривиальный прогноз

Тривиальный прогноз временного ряда X на k шагов вперёд:

```
def prognoz_0(X, k): шагов вперёд
    return np.full(k, X[-1])
```

Качество прогноза

```
def quality_prognoz1(X, k, F):
    '''
    Качество прогнозной функции F на ряде X на k шагов вперё
    :param X: временной ряд
    :param k: на сколько шагов прогнозируем
    :param F: прогнозная функция
    :return:  среднюю погрешность прогноза
    '''
    x_real = X[-k:]        # последние реальные k значений
    x_prog = F(X[:-k], k) # их же прогноз
    return np.linalg.norm(x_real-x_prog)/np.sqr(k)
```

Качество прогноза

```
def quality_prognoz(X, k, F):
    '''
    Качество прогнозной функции F на ряде X на k шагов вперё
    Берем среднее за 60%, 70%, 80%, 90%, 100% от ряда
    :param X: временной ряд
    :param k: на сколько шагов прогнозируем
    :param F: прогнозная функция
    :return:  среднюю погрешнсть прогноза
    '''
    N = len(X)
    errs = []
    for n in (6*N//10, 7*N//10, 8*N//10, 9*N//10, N):
        errs.append(quality_prognoz1(X[:n], k, F))
    #print(np.array(errs))
    return sum(errs)/len(errs)  # средняя ошибка
```