Семинар 10

1. Фильтр Калмана

Решаемая задача: на основе имеющихся наблюдений $\{y_t\}$ и **известных** матриц LSS-системы оценить последовательность векторов состояний $\{x_t\}$.

Предположим, что мы обладаем **всей** информацией на момент времени $t-1$, обозначим ее $y_{\leqslant t-1} \coloneqq y_1, \dots, y_{t-1}$. Если в модели используются нормальные возмущения, то вектор $\begin{pmatrix} y_t \\ x_t \end{pmatrix}$ будет иметь распределение

$$
\begin{aligned}
\begin{pmatrix} y_t \\ x_t \end{pmatrix} &\sim N\left(\begin{pmatrix} y_{t|t-1} \\ x_{t|t-1} \end{pmatrix}; \begin{pmatrix} \Sigma^{yy}_{t|t-1} & \Sigma^{yx}_{t|t-1} \\ \Sigma^{xy}_{t|t-1} & \Sigma^{xx}_{t|t-1} \end{pmatrix} \right) \\
y_{t|t-1} &\coloneqq \mathrm{E}\left(y_t \middle| y_{\leqslant t-1}\right) \\
x_{t|t-1} &\coloneqq \mathrm{E}\left(x_t \middle| y_{\leqslant t-1}\right) \\
\Sigma^{yy}_{t|t-1} &\coloneqq \mathrm{E}\left[\left(y_t - y_{t|t-1}\right)\left(y_t - y_{t|t-1}\right)' \middle| y_{\leqslant t-1} \right] \\
\Sigma^{xx}_{t|t-1} &\coloneqq \mathrm{E}\left[\left(x_t - y_{t|t-1}\right)\left(x_t - x_{t|t-1}\right)' \middle| y_{\leqslant t-1} \right] \\
\Sigma^{yx}_{t|t-1} &\coloneqq \mathrm{E}\left[\left(y_t - y_{t|t-1}\right)\left(x_t - x_{t|t-1}\right)' \middle| y_{\leqslant t-1} \right] \\
(\Sigma^{yx}_{t|t-1})' &= \Sigma^{xy}_{t|t-1}
\end{aligned}
$$

Так как нам известна вся информация на момент времени $t-1$, то нам «известно» значение $x_{t-1|t-1} \coloneqq \mathrm{E}\left(x_{t-1} | y_{t-1}, \dots\right)$.

Фильтр Калмана — итеративная процедура, выполняющая на каждом шаге две операции:

1. прогнозирование — перед появлением нового значения y_t

$$
\begin{aligned}
x_{t|t-1} &= A_t x_{t-1|t-1} \\
y_{t|t-1} &= G_t x_{t|t-1} \\
\Sigma^{xx}_{t|t-1} &= A_t \Sigma^{xx}_{t-1|t-1} A_t' + C_t C_t' \\
\Sigma^{yy}_{t|t-1} &= G_t \Sigma^{xx}_{t|t-1} G_t' + H_t H_t' \\
\Sigma^{yx}_{t|t-1} &= G_t \Sigma^{xx}_{t|t-1}
\end{aligned}
$$

2. коррекция — после его появления

$$
\begin{aligned}
e_t &= y_t - y_{t|t-1} \\
x_{t|t} &= x_{t|t-1} + \left(\Sigma^{yx}_{t|t-1}\right)' \left(\Sigma^{yy}_{t|t-1}\right)^{-1} e_t \\
\Sigma^{xx}_{t|t} &= \Sigma^{xx}_{t|t-1} - \left(\Sigma^{yx}_{t|t-1}\right)' \left(\Sigma^{yy}_{t|t-1}\right)^{-1} \Sigma^{yx}_{t|t-1}
\end{aligned}
$$

Последние две формулы вытекают из многомерного нормального распределения вектора $\begin{pmatrix} y_t \\ x_t \end{pmatrix}$ с плотностью $f_{YX}(y, x)$. Если точнее, то из формулы для плотности условного распределения $f_{X|Y} = \frac{f_{YX}(y,x)}{f_Y(y)}$. На лекции выводилось,кто следил — тот молодец!

Обратите внимание, что матрицы A, C, G, H не обязаны быть постоянными, поэтому в формулах они приведены с индексами времени.

Выполняя эти шаги, мы получим последовательности $\{x_{t|t}\}$ и $\left\{\Sigma^{xx}_{t|t}\right\}$.

1.1. Сглаживание данных

Полученные последовательности можно применить для сглаживания ненаблюдаемых значений x_t. Заметим, что каждое значение $x_{t|t}$ использует только те данные, что доступны на момент времени t. Было бы неплохо иметь возможность иметь оценки вектора состояния, использующие **всю** доступную информацию, то есть $x_{t|T}$.

Для этого воспользуемся сглаживающей схемой Калмана. На первом ее этапе надо прогнать в прямом порядке — от 1 до T — фильтр Калмана и получить последовательности $\{x_{t|t-1}\}, \{x_{t|t}\}, \left\{\Sigma^{xx}_{t|t-1}\right\}$ и $\left\{\Sigma^{xx}_{t|t}\right\}$. Затем надо пройти данные в **обратном** порядке

$$\begin{aligned} J_t &= \Sigma^{xx}_{t|t} A'_{t+1} \left(\Sigma^{xx}_{t+1|t}\right)' \\ x_{t|T} &= x_{t|t} + J_t \left(x_{t+1|T} - x_{t+1|t}\right) \\ \Sigma^{xx}_{t|T} &= \Sigma^{xx}_{t|t} + J_t \left(\Sigma^{xx}_{t+1|T} - \Sigma^{xx}_{t+1|t}\right) J'_t \end{aligned}$$

1.2. Пропущенные наблюдения

Если какое-либо наблюдение y_t пропущено, то при прямом проходе фильтра можно принять $y_t = y_{t|t-1}$, то есть

$$\begin{aligned} x_{t|t} &= x_{t|t-1} \\ \Sigma^{xx}_{t|t} &= \Sigma^{xx}_{t|t-1} \end{aligned}$$

После проведения процедуры сглаживания параметры пропущенного значения можно оценить как

$$\begin{aligned} y_{t|T} &= G_t x_{t|T} \\ \Sigma^{yy}_{t|T} &= G_t \Sigma^{xx}_{t|T} G'_t + H_t H'_t \end{aligned}$$

1.3. Прогнозирование

После прямого прохода можно применить последние значения последовательностей $\{x_{t|t}\}$ и $\left\{\Sigma^{xx}_{t|t}\right\}$ для построения прогнозов $y_t,\ t > T$

$$\begin{aligned} x_{t|t} &= x_{t|t-1} \\ \Sigma^{xx}_{t|t} &= \Sigma^{xx}_{t|t-1} \\ y_{t|T} &= y_{t|t-1} = G x_{t|t-1} \\ \Sigma^{yy}_{t|T} &= \Sigma^{yy}_{t|t-1} = G \Sigma^{xx}_{t|t-1} G' + HH' \end{aligned}$$

2. Эксперименты

```
import numpy as np
import quantecon as qe
import matplotlib.pyplot as plt

plt.rcParams["figure.figsize"] = (11, 6)
```

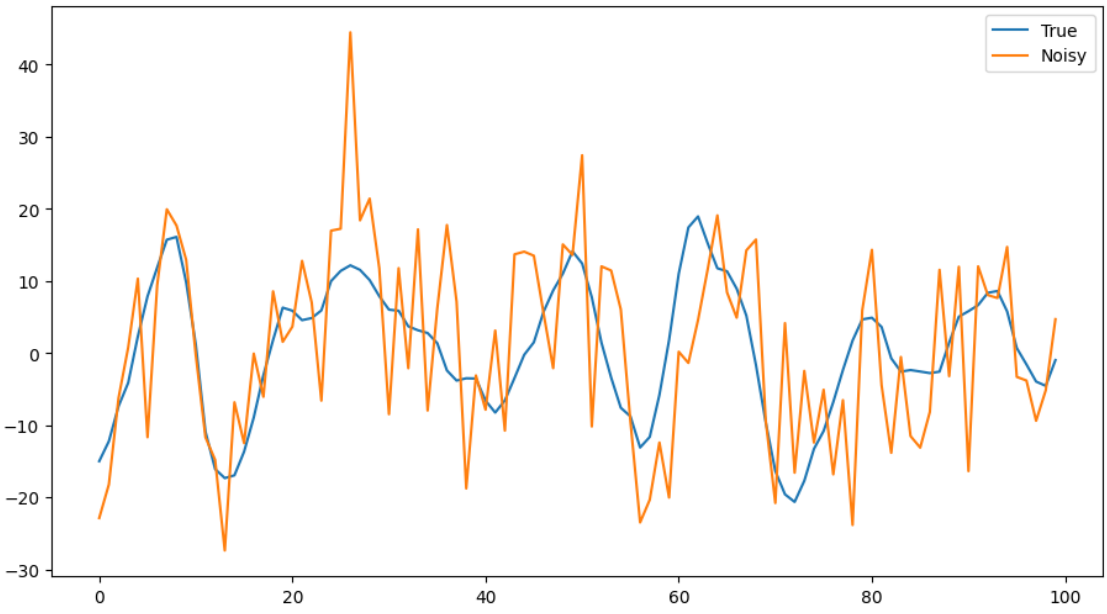
Сгенерируем ряд наблюдений, следующих закону

$$
\begin{aligned}
y_t &= 1.7y_{t-1} - 0.9y_{t-2} + 2\varepsilon_t \\
\tilde{y}_t &= y_t + 10\eta_t \\
y_{-1,-2} &= 0
\end{aligned}
$$

```
n = 100
y = np.zeros(2 * n)
Y = np.zeros(2 * n)

for t in range(2, 2 * n):
    e = np.random.normal(0, 1)
    eta = np.random.normal(0, 1)
    # eta = 0
    y[t] = 1.7 * y[t - 1] - 0.9 * y[t - 2] + 2 * e
    Y[t] = y[t] + 10 * eta
y = y[-n:]
Y = Y[-n:]
```

```
plt.plot(y, label="True")
plt.plot(Y, label="Noisy")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x212a7655010>
```



Так как по предположению фильтра Калмана мы знаем все матрицы, запишем их:

```
A = np.array(
    [
        [1.7, -0.9],
        [1, 0],
    ]
)
C = np.array(
    [
        [2],
        [0],
    ]
)
G = np.array([[1, 0]])
H = np.array([[10]])
```

```
x0 = np.zeros((2, 101))
y0 = np.zeros((1, 101))
xF = np.zeros((2, 101))

Sxx0 = np.zeros((2, 2, 101))
Syy0 = np.zeros((1, 1, 101))
Syx0 = np.zeros((1, 2, 101))
SxxF = np.zeros((2, 2, 101))

SxxF[:, :, 0] = np.identity(2)
```

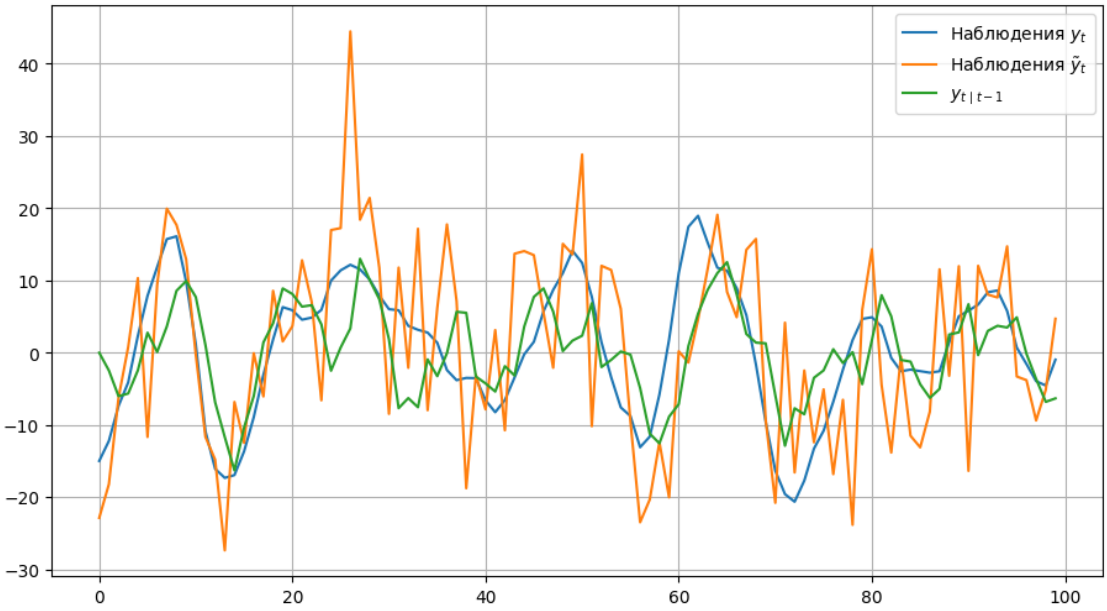
```
for t in range(n):
    # Прогнозирование: x(t|t) -> x(t+1|t)
    x0[:, t + 1] = A @ xF[:, t]
    y0[:, t + 1] = G @ x0[:, t + 1]

    Sxx0[:, :, t + 1] = A @ SxxF[:, :, t] @ A.T + C @ C.T
    Syx0[:, :, t + 1] = G @ Sxx0[:, :, t + 1]
    Syy0[:, :, t + 1] = G @ Sxx0[:, :, t + 1] @ G.T + H @ H.T

    # Коррекция: x(t+1|t) -> x(t+1|t+1)
    e = Y[t] - y0[:, t + 1]  # Внимание на индексы, у нас есть␣
↪начальное значение!

    xF[:, t + 1] = (
        x0[:, t + 1] + Syx0[:, :, t + 1].T @ np.linalg.inv(Syy0[:
↪, :, t + 1]) @ e
    )
    SxxF[:, :, t + 1] = (
        Sxx0[:, :, t + 1]
```

```
        - Syx0[:, :, t + 1].T @ np.linalg.inv(Syy0[:, :, t + 1])␣
↪@ Syx0[:, :, t + 1]
    )
```

```
plt.clf()
fig, (ax) = plt.subplots()
ax.grid()
ax.plot(y, label=r"Наблюдения $y_t$")
ax.plot(Y, label=r"Наблюдения $\tilde{y}_t$")
ax.plot(y0[0, 1:], label=r"$y_{t \mid t-1}$")
ax.legend()
```

```
<matplotlib.legend.Legend at 0x212a99dbd90>
```

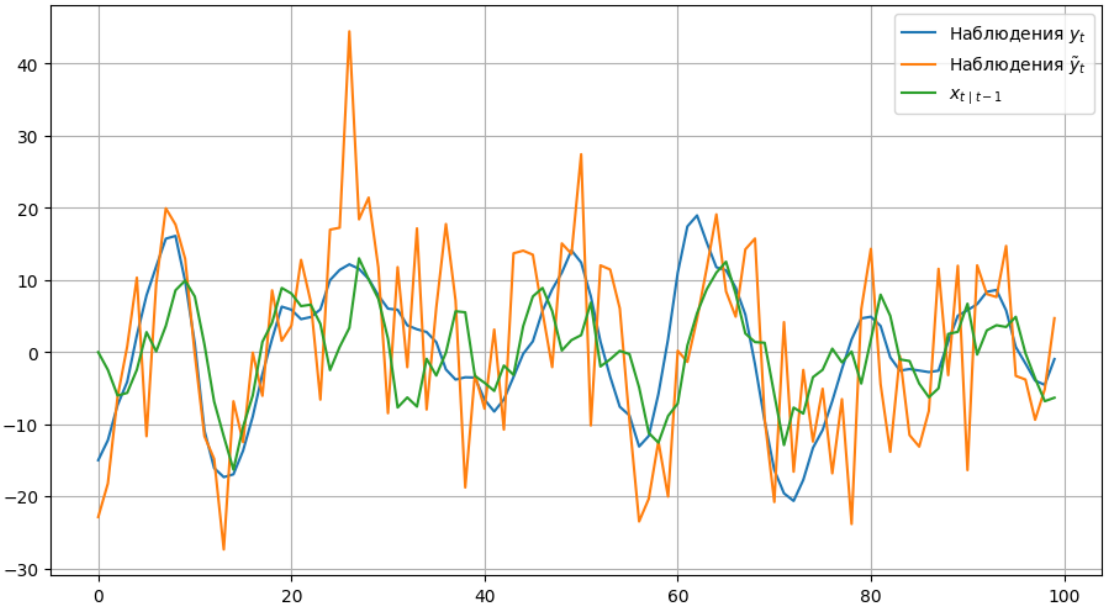
```
<Figure size 1100x600 with 0 Axes>
```




```
plt.clf()
fig, (ax) = plt.subplots()
ax.grid()
ax.plot(y, label=r"Наблюдения $y_t$")
ax.plot(Y, label=r"Наблюдения $\tilde{y}_t$")
ax.plot(x0[0, 1:], label=r"$x_{t \mid t-1}$")
ax.legend()
```

```
<matplotlib.legend.Legend at 0x212a9fd6d50>
```

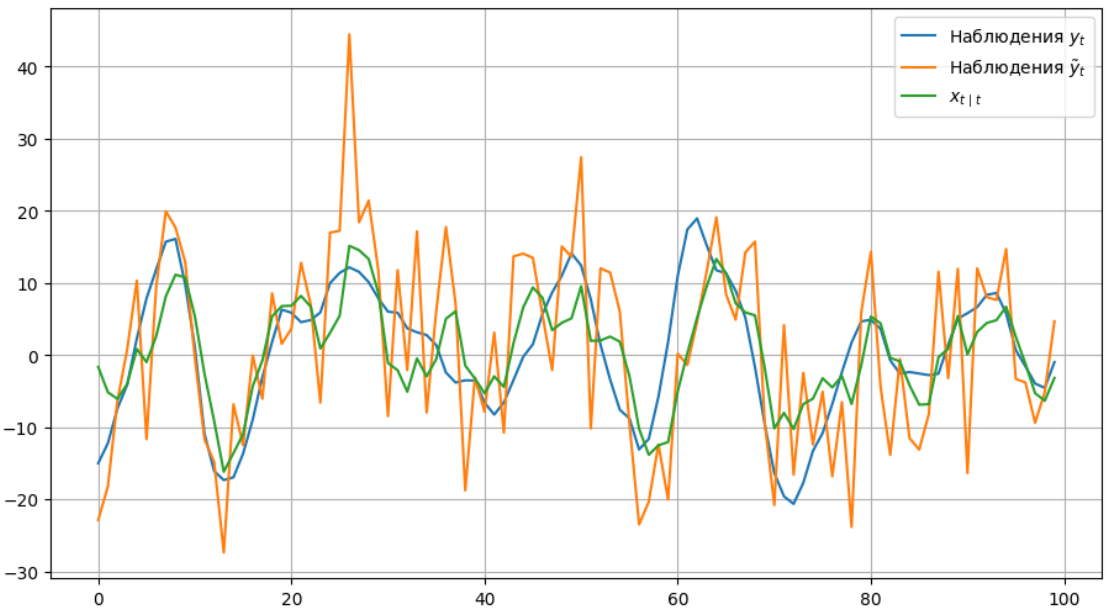
```
<Figure size 1100x600 with 0 Axes>
```




```
plt.clf()
fig, (ax) = plt.subplots()
ax.grid()
ax.plot(y, label=r"Наблюдения $y_t$")
ax.plot(Y, label=r"Наблюдения $\tilde{y}_t$")
ax.plot(xF[0, 1:], label=r"$x_{t \mid t}$")
ax.legend()
```

```
<matplotlib.legend.Legend at 0x212aa27ce10>
```

```
<Figure size 1100x600 with 0 Axes>
```



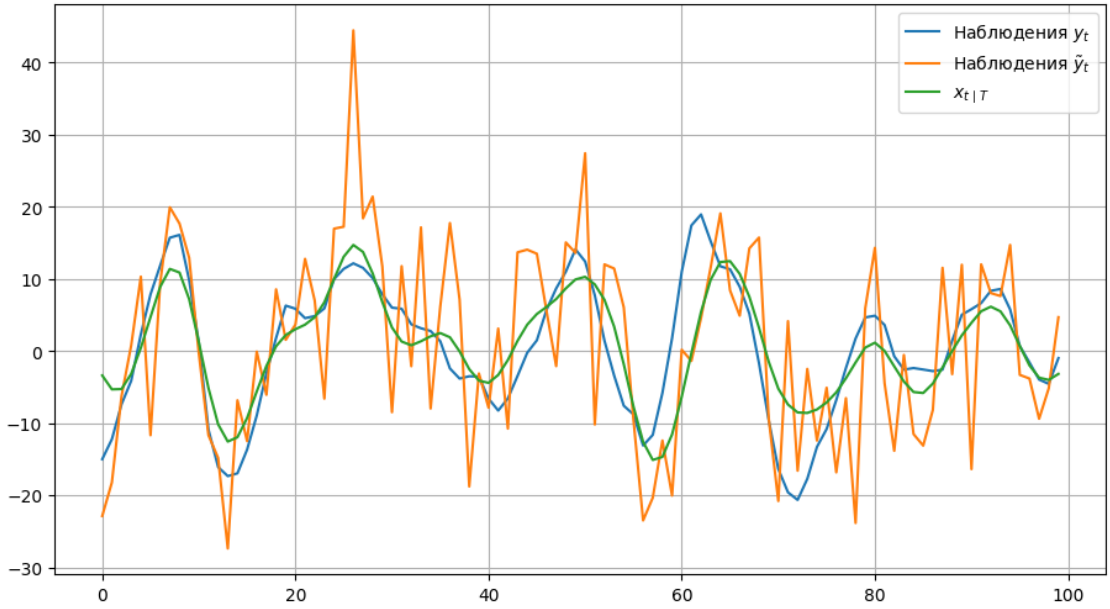
```
xT = np.zeros((2, 101))
SxxT = np.zeros((2, 2, 101))

xT[:, 100] = xF[:, 100]
SxxT[:, :, 100] = SxxF[:, :, 100]
```

```
for t in range(99, 0, -1):
    J = SxxF[:, :, t] @ A.T @ np.linalg.inv(Sxx0[:, :, t + 1])
    xT[:, t] = xF[:, t] + J @ (xT[:, t + 1] - x0[:, t + 1])
    SxxT[:, :, t] = SxxF[:, :, t] + J @ (SxxT[:, :, t + 1] -␣
↪Sxx0[:, :, t + 1]) @ J.T
```

```
plt.clf()
fig, (ax) = plt.subplots()
ax.grid()
ax.plot(y, label=r"Наблюдения $y_t$")
ax.plot(Y, label=r"Наблюдения $\tilde{y}_t$")
ax.plot(xT[0, 1:], label=r"$x_{t \mid T}$")
ax.legend()
```

```
<matplotlib.legend.Legend at 0x212aba16c10>
```

```
<Figure size 1100x600 with 0 Axes>
```



2.1. Долой велосипедостроение

Воспользуемся классом `Kalman`, входящим в пакет `quantecon`. В отличие от класса `LinearStateSpace`, который берет всю работу на себя, `Kalman` может потребовать некоторой ручной работы.

Создаем объект класса `LinearStateSpace`, который затем передаем в класс

Kalman. В качестве начальных значений возьмем вектор из нулей и единичную ковариационную матрицу.

```
ssm = qe.LinearStateSpace(A, C, G, H, mu_0=np.zeros((2, 1)))
kalman = qe.Kalman(ssm, np.zeros((2, 1)), np.identity(2))
```

Класс Kalman содержит в себе два атрибута (помимо прочих): x_hat и Sigma. Первый содержит значения $x_{t|t-1}$ или $x_{t|t}$, второй, соответственно, — $\Sigma^{xx}_{t|t-1}$ или $\Sigma^{xx}_{t|t}$. Класс Kalman также включает в себя несколько методов, из которых нас интересуют три:

- prior_to_filtered. Данный метод принимает на вход одно наблюдение y_t и корректирует значения $x_{t|t-1}$ и $\Sigma^{xx}_{t|t-1}$, превращая их в $x_{t|t}$ и $\Sigma^{xx}_{t|t}$, соответственно.
- filtered_to_forecast. Данный метод строит прогнозы на один шаг вперед, то есть $x_{t+1|t}$ и $\Sigma^{xx}_{t+1|t}$.
- update. Данный метод последовательно выполняет методы prior_to_filtered и filtered_to_forecast, именно в этом порядке. Поэтому при применении этого метода в объекте класса будут храниться **прогнозы**!

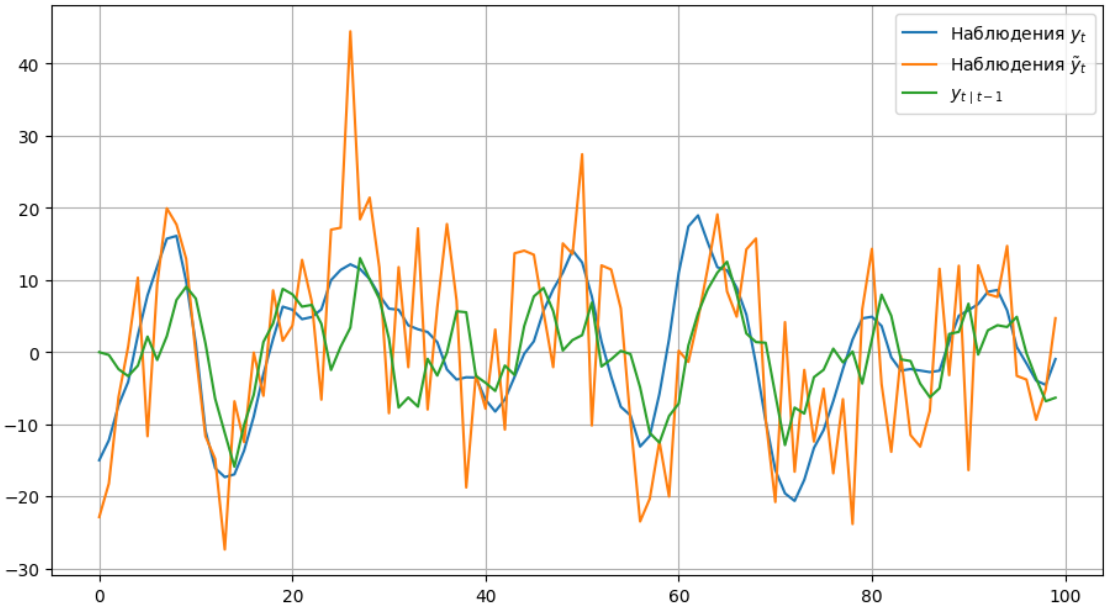
И класс Kalman не умеет в сглаживание и прогнозирование. Он вообще не хранит историю своей работы; все это придется делать вручную. Либо написать класс, наследующий от Kalman и реализующий нужную функциональность.

```
k0 = []
k1 = []

for t in range(n):
    # Сохраняем прогноз y_t
    k0.append(float(kalman.x_hat[0][0]))

    # Корректируем прогнозы при появлении нового наблюдения
    kalman.prior_to_filtered(Y[t])

    # Сохраняем скорректированный прогноз y_t
    k1.append(float(kalman.x_hat[0][0]))

    # Строим следующий прогноз
    kalman.filtered_to_forecast()
```

```
plt.clf()
fig, (ax) = plt.subplots()
ax.grid()
ax.plot(y, label=r"Наблюдения ${y}_t$")
ax.plot(Y, label=r"Наблюдения $\tilde{y}_t$")
ax.plot(k0, label=r"$y_{t \mid t-1}$")
ax.legend()
```

```
<matplotlib.legend.Legend at 0x212aa1ddf90>
```

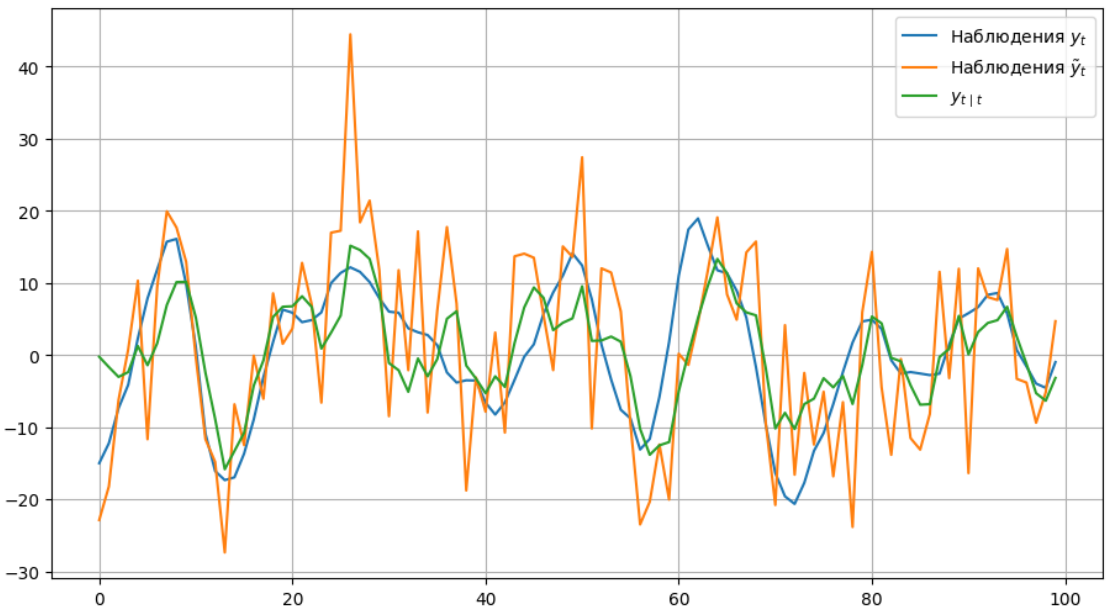
```
<Figure size 1100x600 with 0 Axes>
```




```
plt.clf()
fig, (ax) = plt.subplots()
ax.grid()
ax.plot(y, label=r"Наблюдения ${y}_t$")
ax.plot(Y, label=r"Наблюдения $\tilde{y}_t$")
ax.plot(k1, label=r"$y_{t \mid t}$")
ax.legend()
```

```
<matplotlib.legend.Legend at 0x212ac217d90>
```

```
<Figure size 1100x600 with 0 Axes>
```



2.2. Пример 2

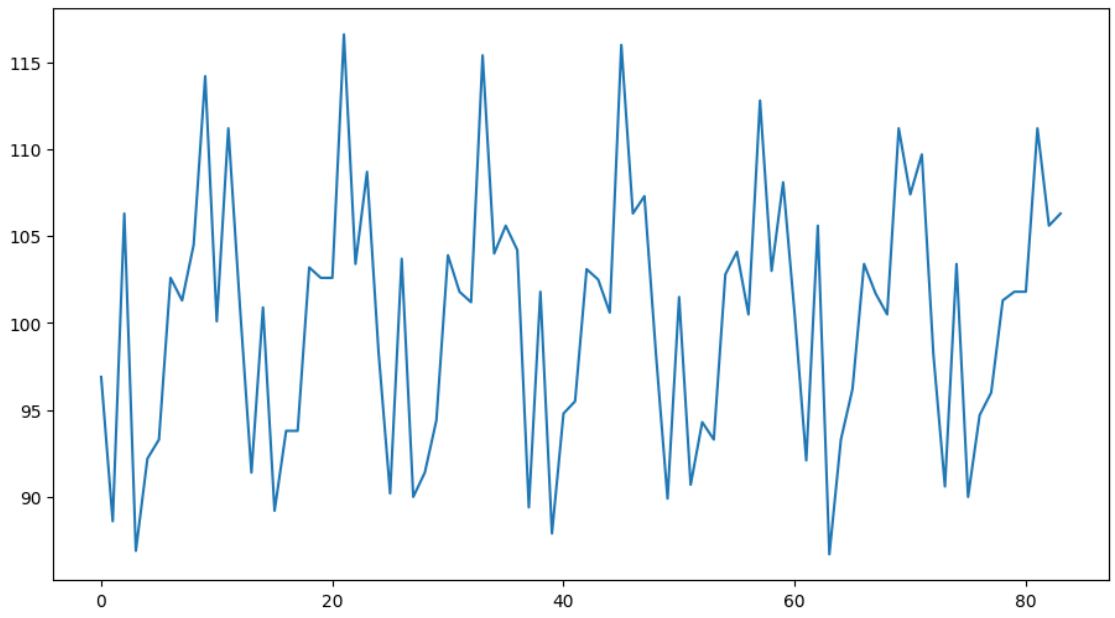
Данные для индекса производства электроэнергии (Росстат).

```
import pandas as pd

data = pd.read_excel("el.xlsx")
y = np.array([data.el])
y
```

```
array([[ 96.9,  88.6, 106.3,  86.9,  92.2,  93.3, 102.6, 101.3,␣
  ↪104.5,
        114.2, 100.1, 111.2, 101. ,  91.4, 100.9,  89.2,  93.8, ␣
  ↪93.8,
        103.2, 102.6, 102.6, 116.6, 103.4, 108.7,  98.3,  90.2,␣
  ↪103.7,
         90. ,  91.4,  94.4, 103.9, 101.8, 101.2, 115.4, 104. ,␣
  ↪105.6,
        104.2,  89.4, 101.8,  87.9,  94.8,  95.5, 103.1, 102.5,␣
  ↪100.6,
        116. , 106.3, 107.3,  98.2,  89.9, 101.5,  90.7,  94.3, ␣
  ↪93.3,
        102.8, 104.1, 100.5, 112.8, 103. , 108.1, 100.5,  92.1,␣
  ↪105.6,
         86.7,  93.3,  96.2, 103.4, 101.7, 100.5, 111.2, 107.4,␣
  ↪109.7,
         98.2,  90.6, 103.4,  90. ,  94.7,  96. , 101.3, 101.8,␣
  ↪101.8,
        111.2, 105.6, 106.3]])
```

```
plt.plot(y[0])
```

```
[<matplotlib.lines.Line2D at 0x212af589090>]
```



2.2.1. Дамми-сезонность

Примем следующую модель данных:

$$\begin{aligned} y_t &= \mu_t + \gamma_t + \varepsilon_t \\ \mu_t &= \mu_{t-1} + \eta_t \\ \gamma_t &= -\sum_{i=1}^{s-1} \gamma_{t-i} + \omega_t \end{aligned}$$

Обратите внимание на определение дамми переменных: их сумма равна 0 с точностью до случайного возмущения. В качестве вектора состояния примем

$$x_t = \begin{pmatrix} \mu_t \\ \gamma_t \\ \vdots \\ \gamma_{t-s+1} \end{pmatrix}$$

Матрицы системы будут равны

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & \dots \\ 0 & -1 & -1 & -1 & -1 & \dots \\ 0 & 1 & 0 & 0 & 0 & \dots \\ 0 & 0 & 1 & 0 & 0 & \dots \\ 0 & 0 & 0 & 1 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}, \quad C = \begin{pmatrix} \sigma_\eta \\ \sigma_\omega \\ 0 \\ 0 \\ \vdots \end{pmatrix}$$

$$G = \begin{pmatrix} 1 & 1 & \underbrace{0 \dots 0}_{10} \end{pmatrix}, \quad H = \begin{pmatrix} \sigma_\varepsilon \end{pmatrix}$$

Начальные условия примем равными

$$\mu_0 = 100, \quad \gamma_{0,-1,\dots,-10} \sim N(0,1), \quad \Sigma^{xx}_{0|0} = I$$

```
s = 12

A = np.zeros((s, s))
A[0, 0] = 1
for t in range(1, s):
    A[1, t] = -1
for t in range(1, s - 1):
    A[t + 1, t] = 1

C = np.zeros((s, 1))
C[0, 0] = 1
C[1, 0] = 1

G = np.zeros((1, s))
G[0, 0] = 1
G[0, 1] = 1
```

```
H = np.array([[1]])

mu_0 = np.zeros((s, 1))
mu_0[0, 0] = 100
mu_0[1:, 0] = np.random.normal(0, 1, size=11)
```

```
A
```

```
array([[ 1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
  ↪0.],
       [ 0., -1., -1., -1., -1., -1., -1., -1., -1., -1., -1., -
1.],
       [ 0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
  ↪0.],
       [ 0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
  ↪0.],
       [ 0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
  ↪0.],
       [ 0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,
  ↪0.],
       [ 0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,
  ↪0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,
  ↪0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,
  ↪0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,
  ↪0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,
  ↪0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,
  ↪0.]])
```

```
ssm = qe.LinearStateSpace(A, C, G, H, mu_0=mu_0)
kalman = qe.Kalman(ssm, mu_0, np.identity(s))
```

```
x0 = np.zeros((s, 84))
xF = np.zeros((s, 84))

Sxx0 = np.zeros((s, s, 84))
SxxF = np.zeros((s, s, 84))

for t in range(84):
    # Сохраняем прогнозы на 1 шаг вперед
    x0[:, t] = kalman.x_hat[:, 0]
    Sxx0[:, :, t] = kalman.Sigma

    # Добавляем новое наблюдение
```

```
    kalman.prior_to_filtered(y[:, t])

    # Сохраняем скорректированные прогнозы на 1 шаг вперед
    xF[:, t] = kalman.x_hat[:, 0]
    SxxF[:, :, t] = kalman.Sigma

    # Прогнозируем на 1 шаг вперед
    kalman.filtered_to_forecast()
```

Выполним сглаживание $x_{t|t}$.

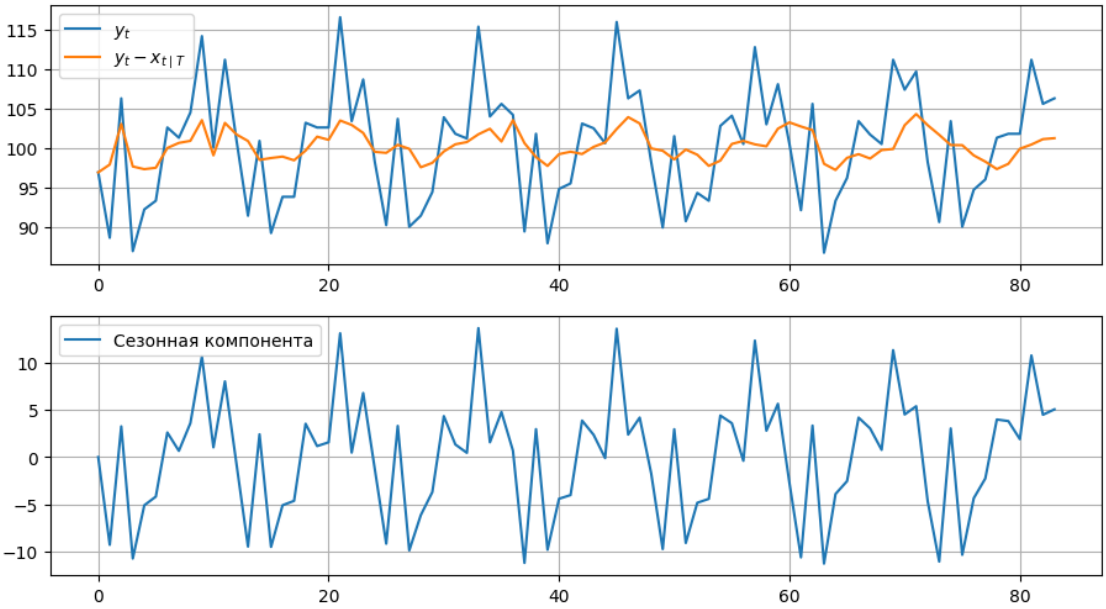
```
xT = np.zeros(xF.shape)
SxxT = np.zeros(SxxF.shape)

xT[:, 83] = xF[:, 83]
SxxT[:, :, 83] = SxxF[:, :, 83]
```

```
for t in range(82, 0, -1):
    J = SxxF[:, :, t] @ A.T @ np.linalg.inv(Sxx0[:, :, t + 1])
    xT[:, t] = xF[:, t] + J @ (xT[:, t + 1] - x0[:, t + 1])
    SxxT[:, :, t] = SxxF[:, :, t] + J @ (SxxT[:, :, t + 1] -␣
  ↪Sxx0[:, :, t + 1]) @ J.T
```

```
yy = []

for t in range(84):
    yy.append(float(y[:, t][0] - xT[1, t]))
```

```
plt.clf()
fig, axes = plt.subplots(2, 1)

axes[0].plot(y[0, :], label=r"$y_t$")
axes[0].plot(yy, label=r"$y_t - x_{t \mid T}$")
axes[1].plot(xT[1, :], label="Сезонная компонента")
for ax in axes:
    ax.grid()
    ax.legend()
```

```
<Figure size 1100x600 with 0 Axes>
```



2.2.2. Тригонометрическая сезонность

Примем следующую модель данных:

$$
\begin{aligned}
y_t &= \mu_t + \gamma_t + \varepsilon_t \\
\mu_t &= \mu_{t-1} + \eta_t \\
\gamma_t &= \sum_{i=1}^{\lfloor s/2 \rfloor} \gamma_{it} \\
\gamma_{it} &= \cos(\lambda_i)\gamma_{i,t-1} + \sin(\lambda_i)\gamma^*_{i,t-1} + \omega_{it}, \qquad i = 1, \dots, \lfloor s/2 \rfloor \\
\gamma^*_{it} &= -\sin(\lambda_i)\gamma_{i,t-1} + \cos(\lambda_i)\gamma^*_{i,t-1} + \omega^*_{it}, \quad i = 1, \dots, \lfloor s/2 \rfloor \\
\lambda_i &= 2\pi\frac{i}{s}
\end{aligned}
$$

В качестве вектора состояния примем

$$
x_t = \begin{pmatrix} \mu_t \\ \gamma_{1t} \\ \vdots \\ \gamma_{6t} \\ \gamma^*_{1t} \\ \vdots \\ \gamma^*_{6t} \end{pmatrix}
$$

Матрицы системы будут равны

$$
A = \begin{pmatrix}
1 & 0 & 0 & \dots & 0 & 0 & \dots \\
0 & \cos(\lambda_1) & 0 & \dots & \sin(\lambda_1) & 0 & \dots \\
0 & 0 & \cos(\lambda_2) & \dots & 0 & \sin(\lambda_2) & \dots \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
0 & -\sin(\lambda_1) & 0 & \dots & \cos(\lambda_1) & 0 & \dots \\
0 & 0 & -\sin(\lambda_2) & \dots & 0 & \cos(\lambda_2) & \dots \\
\vdots & \vdots & \vdots & \dots & \vdots & \vdots & \ddots
\end{pmatrix}, \quad
C = \begin{pmatrix} \sigma_\eta \\ \sigma_{\omega_1} \\ \vdots \\ \sigma_{\omega^*_1} \\ \vdots \end{pmatrix}
$$

$$G = \begin{pmatrix} 1 & \underbrace{1 \dots 1}_{6} & \underbrace{0 \dots 0}_{6} \end{pmatrix}, \quad H = \begin{pmatrix} \sigma_\varepsilon \end{pmatrix}$$

Начальные условия примем равными

$$\mu_0 = 100, \quad \gamma_{1\dots6,0} \sim N(0,1), \quad \gamma^*_{1\dots6,0} \sim N(0,1)$$

$$\sigma_\varepsilon = 1, \quad \Sigma^{xx}_{0|0} = I$$

```
s = 12

A = np.zeros((s + 1, s + 1))
A[0, 0] = 1
for t in range(s // 2):
    A[t + 1, t + 1] = np.cos(2 * np.pi * (t + 1) / 6)
    A[t + 1, t + 7] = np.sin(2 * np.pi * (t + 1) / 6)
    A[t + 7, t + 1] = -np.sin(2 * np.pi * (t + 1) / 6)
    A[t + 7, t + 7] = np.cos(2 * np.pi * (t + 1) / 6)

C = np.zeros((s + 1, 1))
for t in range(s + 1):
    C[t, 0] = 1

G = np.zeros((1, s + 1))
for t in range(7):
    G[0, t] = 1

H = np.array([[1]])

mu_0 = np.zeros((s + 1, 1))
mu_0[0, 0] = 100
mu_0[1:, 0] = np.random.normal(0, 1, size=12)
```

```
np.round(A, 3)
```

```
array([[ 1.   ,  0.   ,  0.   ,  0.   ,  0.   ,  0.   ,  0.   , ␣
  ↪0.   ,
         0.   ,  0.   ,  0.   ,  0.   ,  0.   ],
       [ 0.   ,  0.5  ,  0.   ,  0.   ,  0.   ,  0.   ,  0.   , ␣
  ↪0.866,
         0.   ,  0.   ,  0.   ,  0.   ,  0.   ],
       [ 0.   ,  0.   , -0.5  ,  0.   ,  0.   ,  0.   ,  0.   , ␣
  ↪0.   ,
         0.866,  0.   ,  0.   ,  0.   ,  0.   ],
       [ 0.   ,  0.   ,  0.   , -1.   ,  0.   ,  0.   ,  0.   , ␣
  ↪0.   ,
         0.   ,  0.   ,  0.   ,  0.   ,  0.   ],
       [ 0.   ,  0.   ,  0.   ,  0.   , -0.5  ,  0.   ,  0.   , ␣
  ↪0.   ,
         0.   ,  0.   , -0.866,  0.   ,  0.   ],
       [ 0.   ,  0.   ,  0.   ,  0.   ,  0.   ,  0.5  ,  0.   , ␣
  ↪0.   ,
```

```
        0.   ,  0.   ,  0.   , -0.866,  0.   ],
       [ 0.   ,  0.   ,  0.   ,  0.   ,  0.   ,  0.   ,  1.   , ␣
↪0.   ,
        0.   ,  0.   ,  0.   ,  0.   , -0.   ],
       [ 0.   , -0.866,  0.   ,  0.   ,  0.   ,  0.   ,  0.   , ␣
↪0.5  ,
        0.   ,  0.   ,  0.   ,  0.   ,  0.   ],
       [ 0.   ,  0.   , -0.866,  0.   ,  0.   ,  0.   ,  0.   , ␣
↪0.   ,
       -0.5  ,  0.   ,  0.   ,  0.   ,  0.   ],
       [ 0.   ,  0.   ,  0.   , -0.   ,  0.   ,  0.   ,  0.   , ␣
↪0.   ,
        0.   , -1.   ,  0.   ,  0.   ,  0.   ],
       [ 0.   ,  0.   ,  0.   ,  0.   ,  0.866,  0.   ,  0.   , ␣
↪0.   ,
        0.   ,  0.   , -0.5  ,  0.   ,  0.   ],
       [ 0.   ,  0.   ,  0.   ,  0.   ,  0.   ,  0.866,  0.   , ␣
↪0.   ,
        0.   ,  0.   ,  0.   ,  0.5  ,  0.   ],
       [ 0.   ,  0.   ,  0.   ,  0.   ,  0.   ,  0.   ,  0.   , ␣
↪0.   ,
        0.   ,  0.   ,  0.   ,  0.   ,  1.   ]])
```

```
ssm = qe.LinearStateSpace(A, C, G, H, mu_0=mu_0)
kalman = qe.Kalman(ssm, mu_0, np.identity(s + 1))
```

```
x0 = np.zeros((s + 1, 84))
x0[:, 0] = mu_0[:, 0]

xF = np.zeros((s + 1, 84))

Sxx0 = np.zeros((s + 1, s + 1, 84))
Sxx0[:, :, 0] = np.identity(s + 1) * 0.5
SxxF = np.zeros((s + 1, s + 1, 84))

for t in range(84):
    # Сохраняем прогнозы на 1 шаг вперед
    x0[:, t] = kalman.x_hat[:, 0]
    Sxx0[:, :, t] = kalman.Sigma

    # Добавляем новое наблюдение
    kalman.prior_to_filtered(y[:, t])

    # Сохраняем скорректированные прогнозы на 1 шаг вперед
    xF[:, t] = kalman.x_hat[:, 0]
    SxxF[:, :, t] = kalman.Sigma

    # Прогнозируем на 1 шаг вперед
    kalman.filtered_to_forecast()
```

Выполним сглаживание $x_{t|t}$.

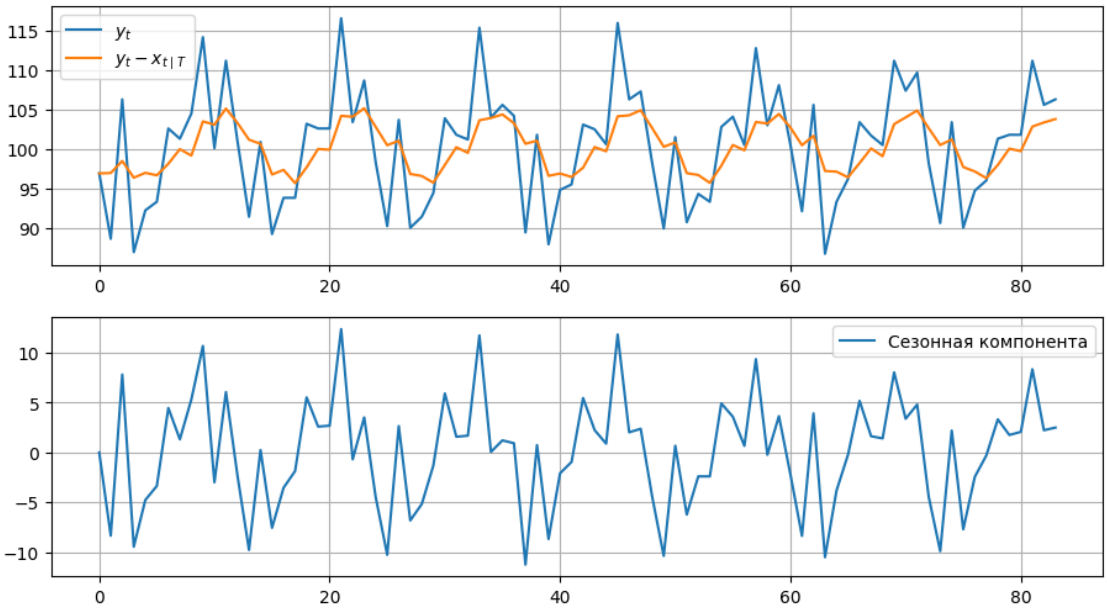
```
xT = np.zeros(xF.shape)
SxxT = np.zeros(SxxF.shape)

xT[:, 83] = xF[:, 83]
SxxT[:, :, 83] = SxxF[:, :, 83]
```

```
for t in range(82, 0, -1):
    J = SxxF[:, :, t] @ A.T @ np.linalg.inv(Sxx0[:, :, t + 1])
    xT[:, t] = xF[:, t] + J @ (xT[:, t + 1] - x0[:, t + 1])
    SxxT[:, :, t] = SxxF[:, :, t] + J @ (SxxT[:, :, t + 1] -␣
  ↪Sxx0[:, :, t + 1]) @ J.T
```

```
yy = []

for t in range(84):
    yy.append(float(y[:, t].item() - sum(xT[1:7, t])))
```

```
plt.clf()
fig, axes = plt.subplots(2, 1)

axes[0].plot(y[0, :], label=r"$y_t$")
axes[0].plot(yy, label=r"$y_t - x_{t \mid T}$")
axes[1].plot([sum(xT[1:7, i]) for i in range(84)],␣
  ↪label="Сезонная компонента")
for ax in axes:
    ax.grid()
    ax.legend()
```

```
<Figure size 1100x600 with 0 Axes>
```



3. Приложения

3.1. Обращение блочной матрицы

Рассмотрим блочную матрицу следующего вида

$$M = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

причем $A_{p\times p}, D_{q\times q}$, значит $M_{(p+q)\times(p+q)}$.

Рассмотрим систему

$$M\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix}$$

Очевидно, что

$$\begin{pmatrix} x \\ y \end{pmatrix} = M^{-1}\begin{pmatrix} u \\ v \end{pmatrix}$$

3.1.1. Вариант 1

Отметим, что перемножение блочных матриц аналогично умножению обычных матриц, что элементарно доказывается «в лоб»

$$\begin{pmatrix} A_1 & B_1 \\ C_1 & D_1 \end{pmatrix}\begin{pmatrix} A_2 & B_2 \\ C_2 & D_2 \end{pmatrix} = \begin{pmatrix} A_1A_2 + B_1C_2 & A_1B_2 + B_1D_2 \\ C_1A_2 + D_1C_2 & C_1B_2 + D_1D_2 \end{pmatrix}$$

Из первого уравнения имеем

$$Ax + By = u$$
$$x = A^{-1}(u - By)$$

Подставив x во второе уравнение, получим

$$CA^{-1}(u - By) + Dy = v$$
$$CA^{-1}u - CA^{-1}By + Dy = v$$
$$y = \left(D - CA^{-1}B\right)^{-1}\left(v - CA^{-1}u\right)$$
$$y = \left[-\left(D - CA^{-1}B\right)^{-1}CA^{-1}\right]u + \left(D - CA^{-1}B\right)^{-1}v$$

Подставив y в выражение для x, получим

$$x = A^{-1}\left[u - B\left(D - CA^{-1}B\right)^{-1}\left(v - CA^{-1}u\right)\right]$$
$$x = \left[A^{-1} + A^{-1}B\left(D - CA^{-1}B\right)^{-1}CA^{-1}\right]u + \left[-A^{-1}B\left(D - CA^{-1}B\right)^{-1}\right]v$$

Итого

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} A^{-1} + A^{-1}B\left(D - CA^{-1}B\right)^{-1}CA^{-1} & -A^{-1}B\left(D - CA^{-1}B\right)^{-1} \\ -\left(D - CA^{-1}B\right)^{-1}CA^{-1} & \left(D - CA^{-1}B\right)^{-1} \end{pmatrix}\begin{pmatrix} u \\ v \end{pmatrix}$$

3.1.2. Вариант 2

Из второго уравнения имеем

$$Cx + Dy = v$$
$$y = D^{-1}(v - Cx)$$

Подставив y в первое уравнение, получим

$$Ax + BD^{-1}(v - Cx) = u$$
$$Ax + BD^{-1}v - BD^{-1}Cx = u$$
$$x = \left(A - BD^{-1}C\right)^{-1}\left(u - BD^{-1}v\right)$$
$$x = \left(A - BD^{-1}C\right)^{-1}u + \left[-\left(A - BD^{-1}C\right)^{-1}BD^{-1}\right]v$$

Подставив x в выражение для y, получим

$$y = D^{-1}\left[v - C\left(A - BD^{-1}C\right)^{-1}\left(u - BD^{-1}v\right)\right]$$
$$y = \left[-D^{-1}C\left(A - BD^{-1}C\right)^{-1}\right]u + \left[D^{-1} + D^{-1}C\left(A - BD^{-1}C\right)^{-1}BD^{-1}\right]v$$

Итого

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \left(A - BD^{-1}C\right)^{-1} & -\left(A - BD^{-1}C\right)^{-1}BD^{-1} \\ -D^{-1}C\left(A - BD^{-1}C\right)^{-1} & D^{-1} + D^{-1}C\left(A - BD^{-1}C\right)^{-1}BD^{-1} \end{pmatrix}\begin{pmatrix} u \\ v \end{pmatrix}$$

3.2. Определитель блочной матрицы.

Известно, что

$$\det(ABC) = \det(A)\det(B)\det(C)$$

Также

$$\det\begin{pmatrix} A & B \\ 0 & D \end{pmatrix} = \det\begin{pmatrix} A & 0 \\ C & D \end{pmatrix} = \det\begin{pmatrix} A & 0 \\ 0 & D \end{pmatrix} = \det(A)\det(D)$$

Рассмотрим блочную матрицу следующего вида

$$M = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

Данную матрицу можно представить в верхнетреугольном виде

$$\begin{pmatrix} I & 0 \\ -CA^{-1} & I \end{pmatrix}\begin{pmatrix} A & B \\ C & D \end{pmatrix} = \begin{pmatrix} A & B \\ 0 & D - CA^{-1}B \end{pmatrix}$$

который приводится к диагональному виду

$$\begin{pmatrix} A & B \\ 0 & D - CA^{-1}B \end{pmatrix}\begin{pmatrix} I & -BA^{-1} \\ 0 & I \end{pmatrix} = \begin{pmatrix} A & 0 \\ 0 & D - CA^{-1}B \end{pmatrix}$$

Итого имеем

$$\begin{pmatrix} I & 0 \\ -CA^{-1} & I \end{pmatrix}\begin{pmatrix} A & B \\ C & D \end{pmatrix}\begin{pmatrix} I & -BA^{-1} \\ 0 & I \end{pmatrix} = \begin{pmatrix} A & 0 \\ 0 & D - CA^{-1}B \end{pmatrix}$$

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} = \begin{pmatrix} I & 0 \\ CA^{-1} & I \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & D - CA^{-1}B \end{pmatrix} \begin{pmatrix} I & BA^{-1} \\ 0 & I \end{pmatrix}$$

или, аналогично,

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} = \underbrace{\begin{pmatrix} I & BD^{-1} \\ 0 & I \end{pmatrix}}_{\det(\cdot)=1} \begin{pmatrix} A - BD^{-1}C & 0 \\ 0 & D \end{pmatrix} \underbrace{\begin{pmatrix} I & 0 \\ D^{-1}C & I \end{pmatrix}}_{\det(\cdot)=1}$$

Отсюда

$$\det \begin{pmatrix} A & B \\ C & D \end{pmatrix} = \det(A) \det(D - CA^{-1}B) = \det(A - BD^{-1}C) \det(D)$$

3.3. Вывод формул коррекции

Вспомним, что

$$\underbrace{\begin{pmatrix} y_t \\ x_t \end{pmatrix}}_{Z} \sim N\left(\underbrace{\begin{pmatrix} y_{t|t-1} \\ x_{t|t-1} \end{pmatrix}}_{\mu}; \underbrace{\begin{pmatrix} \Sigma^{yy}_{t|t-1} & \Sigma^{yx}_{t|t-1} \\ \Sigma^{xy}_{t|t-1} & \Sigma^{xx}_{t|t-1} \end{pmatrix}}_{\Sigma} \right)$$

Формула для плотности многомерного нормального распределения имеет вид

$$f_{YX}(Z \mid \mu, \Sigma) = \frac{1}{\sqrt{2\pi}|\Sigma|^{1/2}} e^{-\frac{1}{2}(Z-\mu)'\Sigma^{-1}(Z-\mu)}$$

Плотность условного распределения $x \mid y = y_t$ равна

$$\begin{aligned} f_{X|Y}(x \mid x_{t|t}, \Sigma^{xx}_{t|t}) = \frac{f_{YX}(Z \mid \mu, \Sigma)}{f_Y(y_t \mid y_{t|t-1}, \Sigma^{yy}_{t|t-1})} = \\ = \frac{|\Sigma^{yy}_{t|t-1}|^{1/2}}{|\Sigma|^{1/2}} e^{-\frac{1}{2}\left[(Z-\mu)'\Sigma^{-1}(Z-\mu) - (y_t - y_{t|t-1})'(\Sigma^{yy}_{t|t-1})^{-1}(y_t - y_{t|t-1})\right]} \end{aligned}$$

где $x_{t|t} = \mathrm{E}(x \mid y = y_t)$, $\Sigma^{xx}_{t|t} = \mathrm{Var}(x \mid y = y_t)$.

Первый множитель преобразуется в

$$\frac{|\Sigma^{yy}_{t|t-1}|^{1/2}}{|\Sigma|^{1/2}} \Rightarrow \frac{1}{\sqrt{\det\left(\Sigma^{xx}_{t|t-1} - \Sigma^{xy}_{t|t-1}(\Sigma^{yy}_{t|t-1})^{-1}\Sigma^{yx}_{t|t-1}\right)}} = \frac{1}{\sqrt{\det \Sigma^{xx}_{t|t}}}$$

Для преобразования показателя экспоненты будем считать x_t и y_t (пока!) центрированными. Также отбросим нижний индекс $(t \mid t-1)$, так как других не наблюдается. Это сократит визуальный шум.

Получим

$$
\begin{aligned}
Z'\Sigma^{-1}Z - y_t'(\Sigma^{yy})^{-1}y_t = \\
&= \begin{pmatrix} y_t' & x_t' \end{pmatrix}' \begin{pmatrix} \Sigma^{yy} & \Sigma^{yx} \\ \Sigma^{xy} & \Sigma^{xx} \end{pmatrix}^{-1} \begin{pmatrix} y_t \\ x_t \end{pmatrix} - y_t'(\Sigma^{yy})^{-1}y_t = \\
&= \begin{pmatrix} y_t' & x_t' \end{pmatrix}' \begin{pmatrix} (\Sigma^{yy})^{-1} + (\Sigma^{yy})^{-1}\Sigma^{yx}\hat{\Sigma}^{-1}\Sigma^{xy}(\Sigma^{yy})^{-1} & -(\Sigma^{yy})^{-1}\Sigma^{yx}\hat{\Sigma}^{-1} \\ -\hat{\Sigma}^{-1}\Sigma^{xy}(\Sigma^{yy})^{-1} & \hat{\Sigma}^{-1} \end{pmatrix} \begin{pmatrix} y_t \\ x_t \end{pmatrix} - \\
&\quad - y_t'(\Sigma^{yy})^{-1}y_t = \\
&= y_t'(\Sigma^{yy})^{-1}y_t + y_t'(\Sigma^{yy})^{-1}\Sigma^{yx}\hat{\Sigma}^{-1}\Sigma^{xy}(\Sigma^{yy})^{-1}y_t - x_t'\hat{\Sigma}^{-1}\Sigma^{xy}(\Sigma^{yy})^{-1}y_t - \\
&\quad - y_t'(\Sigma^{yy})^{-1}\Sigma^{yx}\hat{\Sigma}^{-1}x_t + x_t'\hat{\Sigma}^{-1}x_t - y_t'(\Sigma^{yy})^{-1}y_t = \\
&= x_t'\hat{\Sigma}^{-1}\left(x_t - \Sigma^{xy}(\Sigma^{yy})^{-1}y_t\right) - y_t'(\Sigma^{yy})^{-1}\Sigma^{yx}\hat{\Sigma}^{-1}\left(x_t - \Sigma^{xy}(\Sigma^{yy})^{-1}y_t\right) = \\
&= \left(x_t' - y_t'(\Sigma^{yy})^{-1}\Sigma^{yx}\right)\hat{\Sigma}^{-1}\left(x_t - \Sigma^{xy}(\Sigma^{yy})^{-1}y_t\right) = \\
&= \left(x_t - \Sigma^{xy}(\Sigma^{yy})^{-1}y_t\right)'\hat{\Sigma}^{-1}\left(x_t - \Sigma^{xy}(\Sigma^{yy})^{-1}y_t\right)
\end{aligned}
$$

Если вспомнить формулу для многомерного нормального распределения и провести обратные подстановки

$$
\begin{aligned}
x_t &\Rightarrow x_t - x_{t|t-1} \\
y_t &\Rightarrow y_t - y_{t|t-1} = e_t
\end{aligned}
$$

то станет ясно, что

$$
x_t := \mathrm{E}(x \mid y = y_t) = x_{t|t-1} + \Sigma^{xy}(\Sigma^{yy})^{-1}e_t
$$