In [1]:

```python
# generate oryx data based on scraped data from https://github.com/leedra
from glob import glob
import csv
import os

def generate_oryx_data():
    if os.path.exists('oryx.csv'):
        return

    last_data = {}

    result = []
    columns = set()

    for filename in sorted(glob('data/*.csv'), reverse=True):
        with open(filename, 'r', encoding='utf-8') as f:
            c = csv.reader(f)
            header = next(c)

            today_losses = {}

            for row in c:
                row = dict(zip(header, row))

                # clean data
                if row['equipment_type'] == 'All Types':
                    continue
                if row['equipment_type'].startswith('Losses of Armoured C
                    continue
                if row['equipment_type'].startswith('Losses excluding Rec
                    continue

                # merge certain columns
                if row['equipment_type'].startswith('Naval Ships'):
                    row['equipment_type'] = 'Naval Ships'

                if row['equipment_type'].startswith('Trucks, Vehicles'):
                    row['equipment_type'] = 'Trucks, Vehicles and Jeeps'

                if 'Communication' in row['equipment_type']:
                    row['equipment_type'] = 'Command Posts And Communicat

                if row['country'] != 'Russia':
                    continue

                row['equipment_type'] = row['equipment_type'].lower().str

                if row['equipment_type'] not in last_data:
                    last_data[row['equipment_type']] = int(row['type_tota

                # calculate difference
                today_losses[row['equipment_type']] = last_data[row['equi
                columns.add(row['equipment_type'])
                today_losses['date'] = row['Date']
```

```
                last_data[row['equipment_type']] = int(row['type_total'])

            result.append(today_losses)


    for item in result:
        for column in columns:
            if column not in item:
                item[column] = 0


    result = result[1:]

    header = ['date', 'armoured fighting vehicles', 'armoured personnel c

    with open('oryx.csv', 'w', encoding='utf-8') as f:
        c = csv.writer(f)
        c.writerow(header)
        for row in result[::-1]:
            c.writerow(map(str, [row[x] for x in header]))

generate_oryx_data()
```

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import numpy as np

data = pd.read_csv('oryx.csv', dtype={
    "date": str,
    "armoured fighting vehicles": np.float64,
    "armoured personnel carriers": np.float64,
    "infantry fighting vehicles": np.float64,
    "infantry mobility vehicles": np.float64,
    "tanks": np.float64,
    "trucks, vehicles and jeeps": np.float64
})
data['date'] = pd.to_datetime(data['date'], format='%Y-%m-%d')
data = data.fillna(0)
```

In [3]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

data = pd.read_csv('oryx.csv', dtype={
    "date": str,
    "armoured fighting vehicles": np.float64,
    "armoured personnel carriers": np.float64,
    "infantry fighting vehicles": np.float64,
    "infantry mobility vehicles": np.float64,
    "tanks": np.float64,
    "trucks, vehicles and jeeps": np.float64
})
data['date'] = pd.to_datetime(data['date'], format='%Y-%m-%d')
data = data.fillna(0)
```

4.1.1 Формування файлу даних у формі таблиць.

In [4]:
```
data.head()
```

Out[4]:

	date	armoured fighting vehicles	armoured personnel carriers	infantry fighting vehicles	infantry mobility vehicles	tanks	trucks, vehicles and jeeps
0	2022-02-24	2.0	0.0	3.0	3.0	1.0	10.0
1	2022-02-25	0.0	0.0	7.0	2.0	5.0	18.0
2	2022-02-26	4.0	0.0	4.0	4.0	2.0	29.0
3	2022-02-27	20.0	8.0	28.0	9.0	32.0	42.0
4	2022-02-28	5.0	0.0	5.0	2.0	3.0	9.0

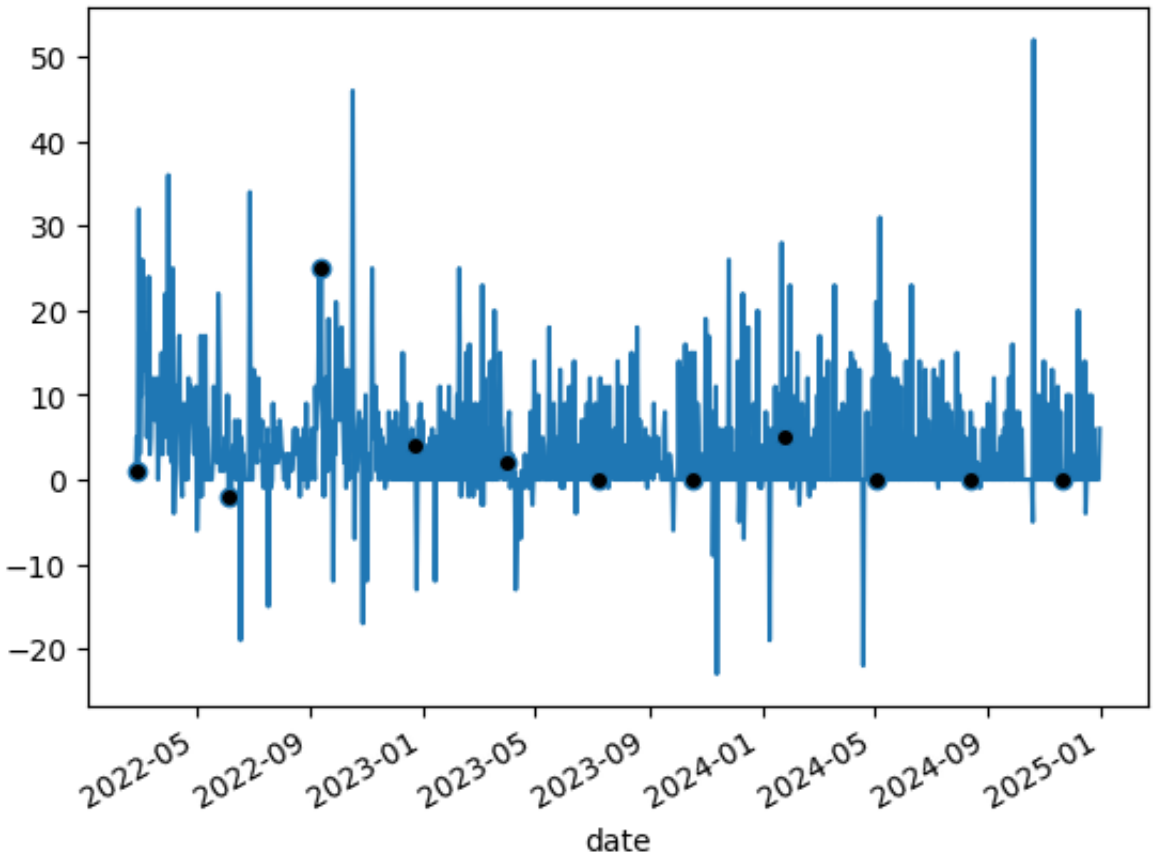
In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1041 entries, 0 to 1040
Data columns (total 7 columns):
 #   Column                        Non-Null Count  Dtype         
---  ------                        --------------  -----         
 0   date                          1041 non-null   datetime64[ns]
 1   armoured fighting vehicles    1041 non-null   float64       
 2   armoured personnel carriers   1041 non-null   float64       
 3   infantry fighting vehicles    1041 non-null   float64       
 4   infantry mobility vehicles    1041 non-null   float64       
 5   tanks                         1041 non-null   float64       
 6   trucks, vehicles and jeeps    1041 non-null   float64       
dtypes: datetime64[ns](1), float64(6)
memory usage: 57.1 KB
```

4.1.2. Графічне подання даних.

Графіки даних в декартовій системі координат

In [6]:
```
data.set_index('date', inplace=True)
plot = data['tanks'].plot(linestyle='-', markevery=100, marker='o', marke
```



Часовий ряд:

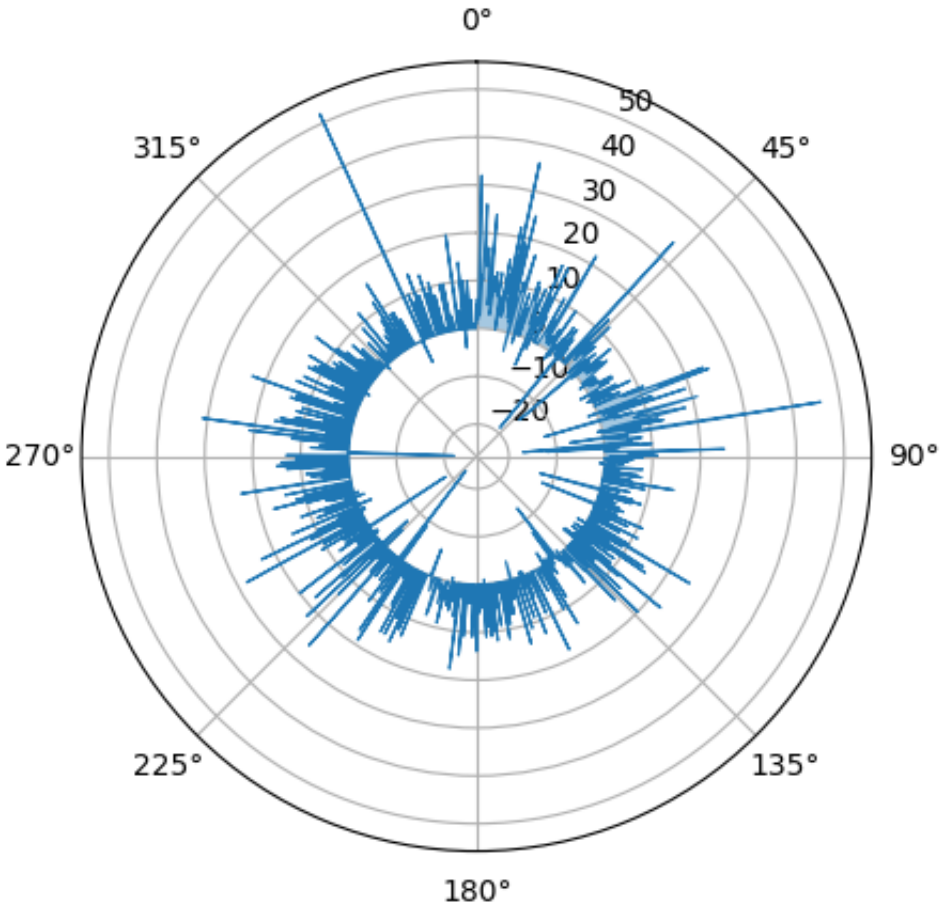
Значення: Графік відображає зміну значення з часом. Видно сильну волатильність, але є тенденція до стабілізації. Позначення кожного 100-го значення допомагає відмітити ключові точки.

Висновок: Дані досить нерівномірні, з піками в деякі періоди. Це вказує на сезонність або аномальні події щодо знищення танків русні.

In [7]:
```
# ig, axis = plt.subplots(3, 2, figsize=(8,10))
# hist = data.hist(grid=False, ax=axis, bins=int(1 + np.log2(data.shape[0
```

Графіки даних в полярній системі координат

In [8]:
```
ax = plt.subplot(projection='polar')
ax.set_theta_direction(-1)
ax.set_theta_zero_location("N")

t = mdates.date2num(data.index.to_pydatetime())
y = data['tanks']
tnorm = (t-t.min())/(t.max()-t.min())*2.*np.pi
ax.fill_between(tnorm,y ,0, alpha=0.4)
ax.plot(tnorm,y , linewidth=0.8)
plt.show()
```



4.2. Описова статистика – кількісні характеристики даних.

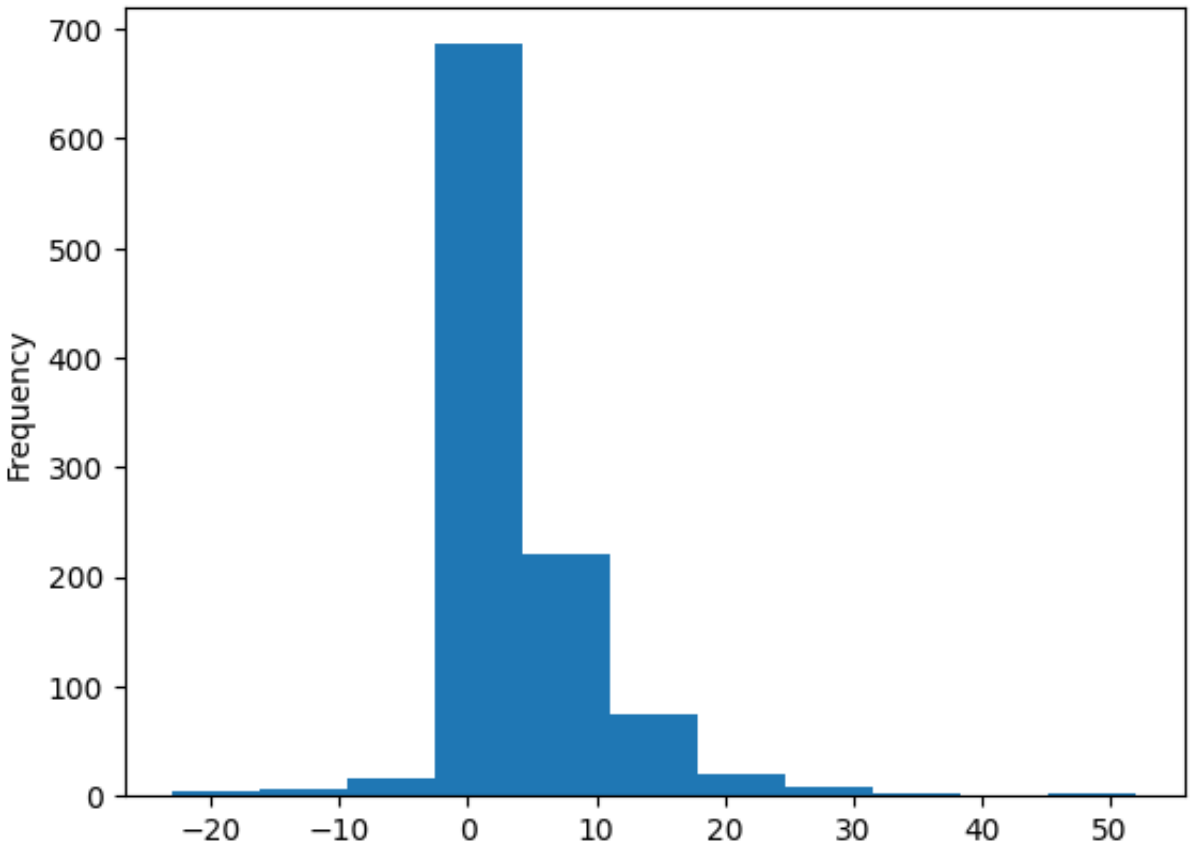
In [9]:
```
print("Результати описової статистики")
print("Показники/Значення")
print("Середнє: %s" % data.tanks.mean())
print("Стандартна помилка: %s" % (data.tanks.std() / data.tanks.count() *
print("Медіана: %s" % data.tanks.median())
print("Мода: %s" % data.tanks.mode()[0])
print("Стандартне відхилення: %s" % data.tanks.std())
print("Дисперсія вибірки: %s" % data.tanks.std() ** 2)
print("Ексцес: %s" % (data.tanks.kurt() - 3))
print("Асиметричність: %s" % data.tanks.skew())
print("Інтервал: %s" % (data.tanks.max() - data.tanks.min()))
print("Мінімум: %s" % data.tanks.min())
print("Максимум: %s" % data.tanks.max())
print("Сума: %s" % data.tanks.sum())
print("Обсяг: %s" % data.tanks.count())
print("Рівень надійності (95,0%%): %s" % data.tanks.quantile(q=0.95))
```

```
Результати описової статистики
Показники/Значення
Середнє: 3.5225744476464937
Стандартна помилка: 0.1981443310374142
Медіана: 0.0
Мода: 0.0
Стандартне відхилення: 6.39303403205999
Дисперсія вибірки: 40.87088413507722
Ексцес: 5.04857559691785
Асиметричність: 1.720186287832713
Інтервал: 75.0
Мінімум: -23.0
Максимум: 52.0
Сума: 3667.0
Обсяг: 1041
Рівень надійності (95,0%): 15.0
```

4.2.1. Побудова гістограми

In [10]:
```
# формула Стерджеса
k = int(1 + np.log2(data.shape[0]))
```

In [11]:
```
plot = data['tanks'].plot.hist(bins=k)
```



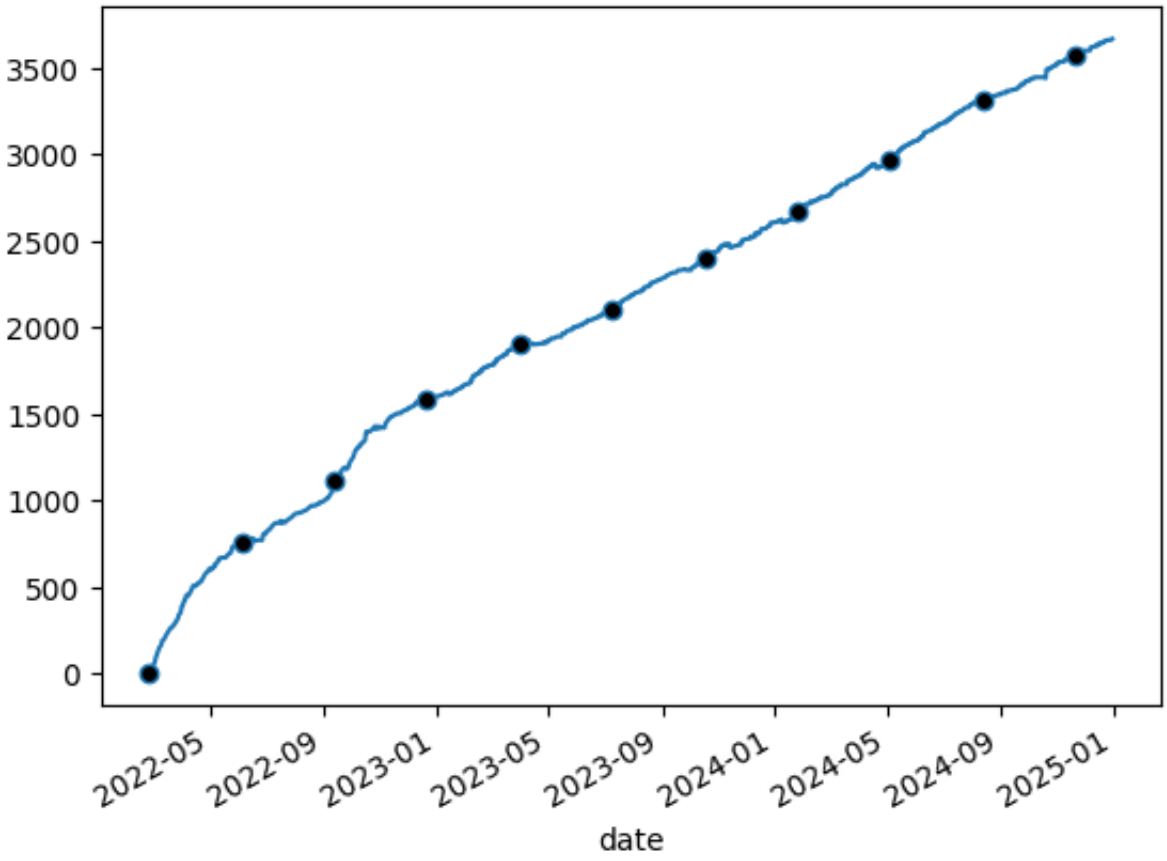
Гістограма:

Значення: Це розподіл значень змінної tanks. Більшість даних зосереджені близько нуля, але є невелика кількість значень з більшими відхиленнями.
Висновок: Розподіл є правобічним, з кількома аномальними значеннями. Більшість спостережень мають значення від 0 до 10.

4.2.2. Побудова кумуляти.

Усі значення додаються одне до одного в послідовності, створюючи наростаючий результат.

In [12]:
```
plot = data['tanks'].cumsum().plot(linestyle='-', markevery=100, marker='
```



Кумулятивний графік:

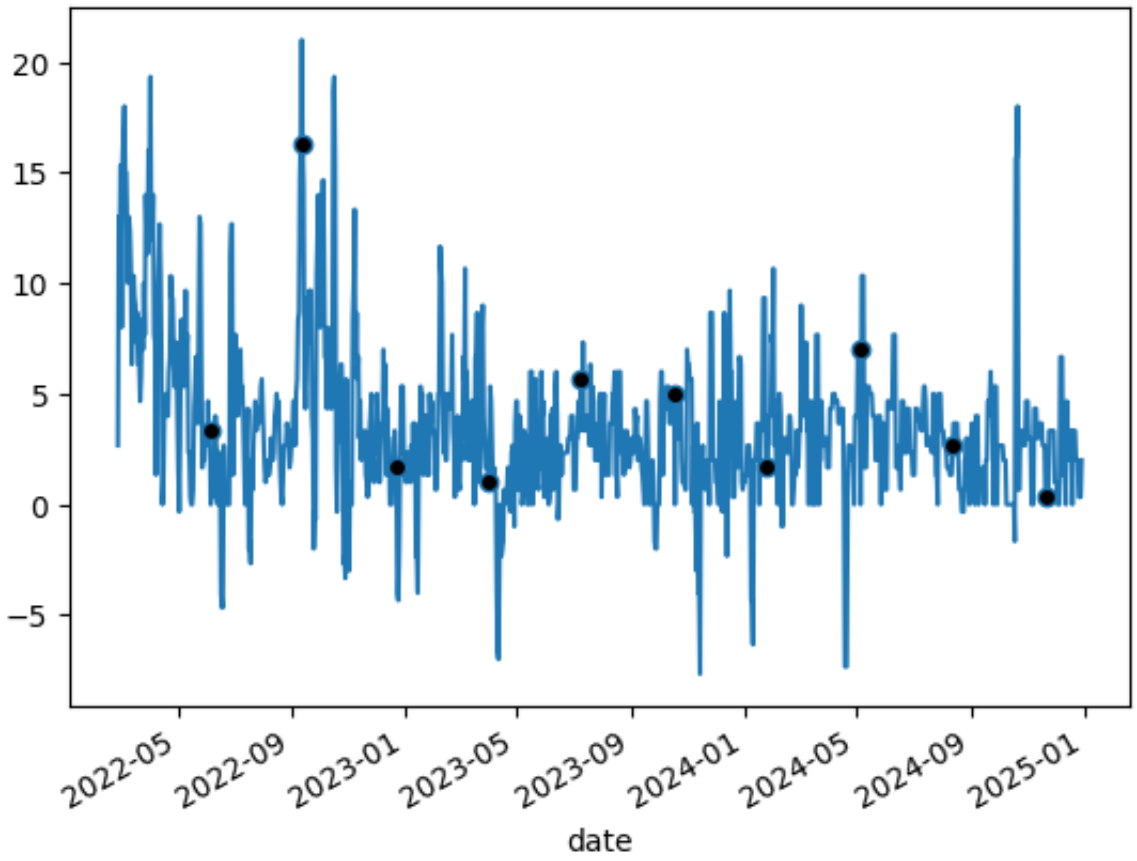
Значення: Відображає накопичену суму значень змінної tanks з часом. Графік лінійний, що свідчить про стабільний тренд.

Висновок: Дані демонструють постійне зростання без суттєвих збоїв чи різких змін.

Лінійне згладжування По суті, беруться дані з вікном n і визначається середнє значення цих n даних, в результаті получаємо більш згладжений і рівний графік.

Наприклад 3 вікна, для значень [x1, x2, x3] середнє обчислюється як (x1 + x2 + x3) / 3 і ставиться на місце x2.

In [13]:
```
# Лінійне згладжування для 3
plot = data.tanks.rolling(window=3, center=True).mean().plot(linestyle='-
```



Згладжений часовий ряд (ковзне середнє):

Значення: Графік показує ковзне середнє (із вікном 3), яке згладжує коливання. Це дозволяє побачити загальний вигляд.

Висновок: Є загальна тенденція до стабілізації, але локальні коливання все ще помітні.

5. Виявлення тенденції часового ряду методами згладжування

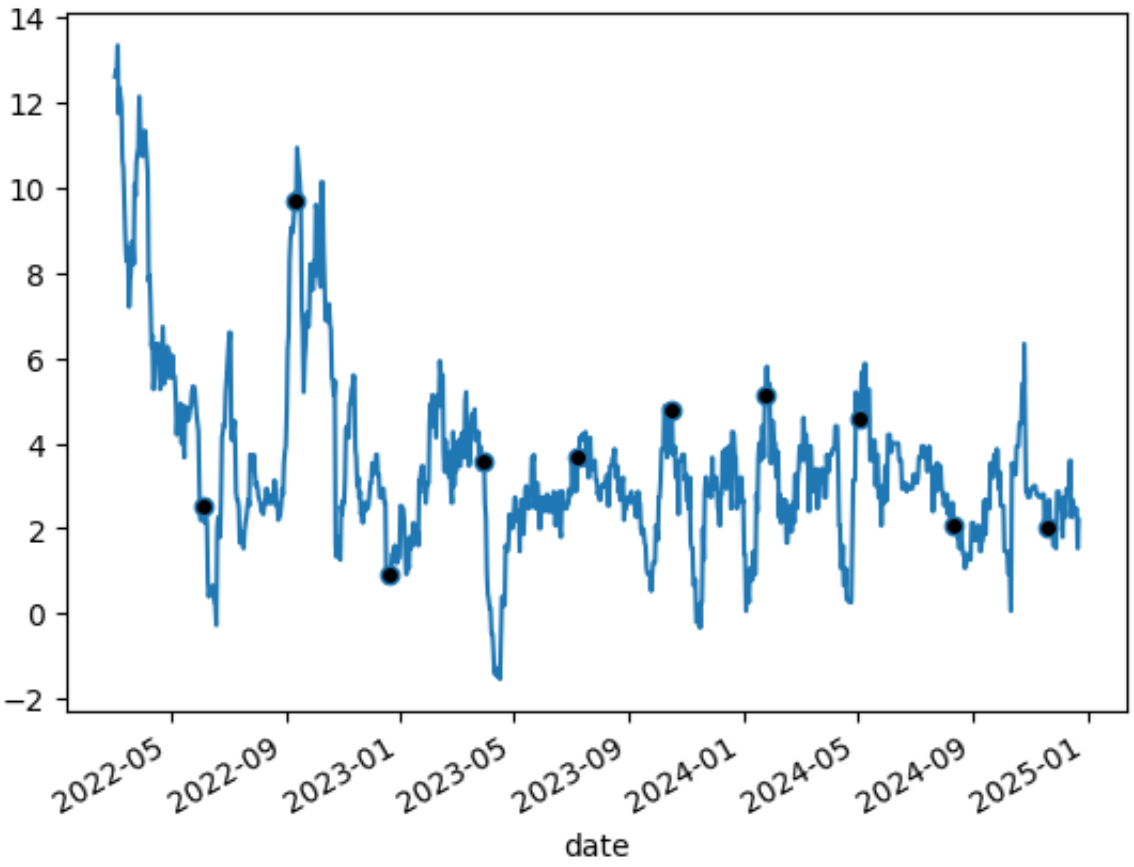
5.2 Метод ковзної середньої

In [14]:
```
# Лінійне згладжування для 3
series3 = data.tanks.rolling(window=3, center=True).mean()
```

In [15]:
```
# Лінійне згладжування для 5
series5 = data.tanks.rolling(window=5, center=True).mean()
```

In [16]:
```
# Лінійне згладжування для 7
series7 = data.tanks.rolling(window=7, center=True).mean()
```

In [17]:
```
# Лінійне згладжування для 15
plot = data.tanks.rolling(window=15, center=True).mean().plot(linestyle='
```



Різні згладження (вікна 3, 5, 7, 15):

Значення: Графіки порівнюють ефекти різних інтервалів згладження. При збільшенні розміру вікна (до 15) графік стає більш плавним.

Висновок: Використання ширшого вікна допомагає виявити довгострокові тренди, але приховує короткострокові коливання.

5.3 Метод зваженої ковзної середньої за формулами Кендела

In [18]:
```
# 3
weights = [
    np.array([5, 2, -1]) / 6,
    np.array([1, 1, 1]) / 3,
    np.array([-1, 2, 5]) / 6
]
print(weights)
series3 = data.tanks.rolling(window=3, center=True).apply(lambda x: np.su

series3[series3.index[0]] = np.sum(data.tanks[:3] * weights[0])
series3[series3.index[-1]] = np.sum(data.tanks[-3:] * weights[-1])
```

```
[array([ 0.83333333,  0.33333333, -0.16666667]), array([0.33333333, 0.3333
3333, 0.33333333]), array([-0.16666667,  0.33333333,  0.83333333])]
```

In [19]:
```
# 5
weights = [
    np.array(list(range(3, -2, -1))) / 5,
```

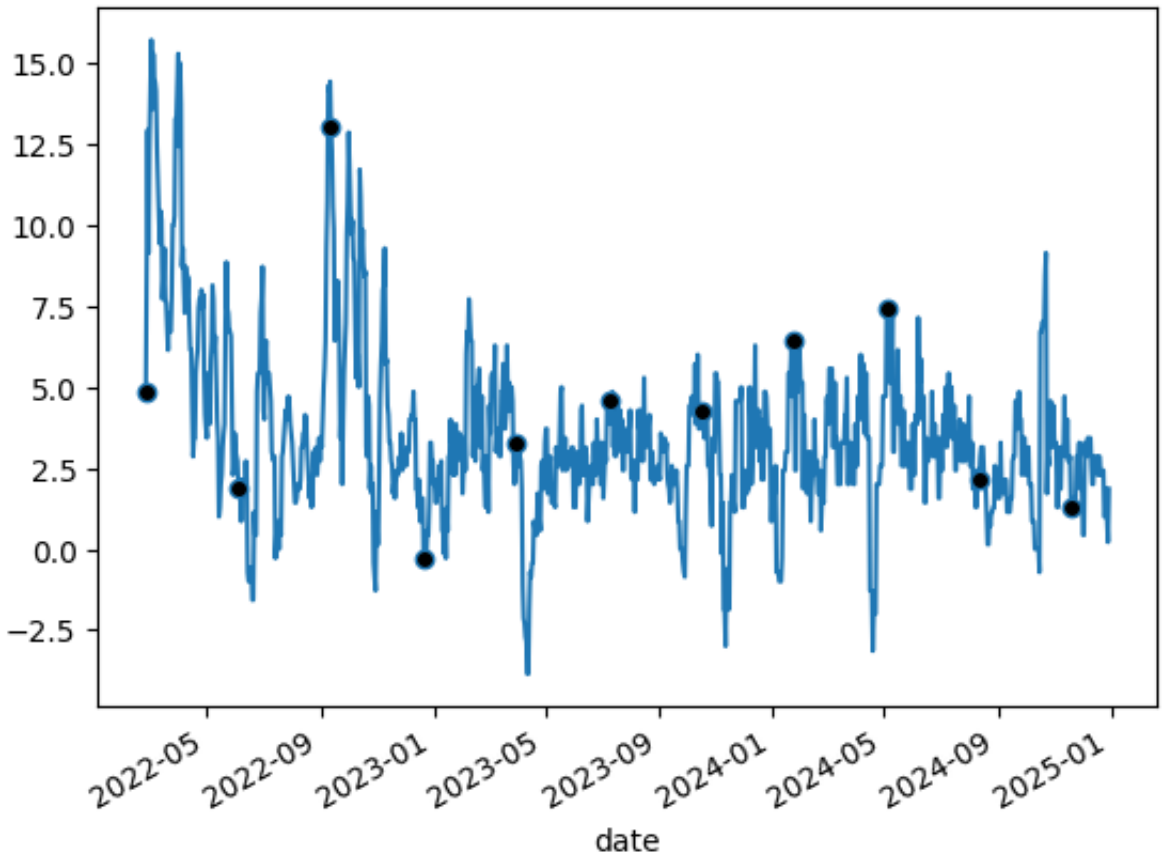
```
    np.array(list(range(5, 0, -1))) / 10,
    np.array([1] * 5) / 5,
    np.array(list(range(1, 6))) / 10,
    np.array(list(range(-1, 4))) / 5,
]
print(weights)
series5 = data.tanks.rolling(window=5, center=True).apply(lambda x: np.su

series5[series5.index[0]] = np.sum(data.tanks[:5] * weights[0])
series5[series5.index[1]] = np.sum(data.tanks[1:6] * weights[1])
series5[series5.index[-2]] = np.sum(data.tanks[-6:-1] * weights[-2])
series5[series5.index[-1]] = np.sum(data.tanks[-5:] * weights[-1])
```

```
[array([ 0.6,  0.4,  0.2,  0. , -0.2]), array([0.5, 0.4, 0.3, 0.2, 0.1]),
array([0.2, 0.2, 0.2, 0.2, 0.2]), array([0.1, 0.2, 0.3, 0.4, 0.5]), arra
y([-0.2,  0. ,  0.2,  0.4,  0.6])]
```

In [20]:

```
# 7
weights = [
    np.array(list(range(13, -6, -3))) / 28,
    np.array(list(range(5, -2, -1))) / 14,
    np.array(list(range(7, 0, -1))) / 28,
    np.array([1] * 7) / 7,
    np.array(list(range(1, 8))) / 28,
    np.array(list(range(-1, 6))) / 14,
    np.array(list(range(-5, 14, 3))) / 28,
]
print(weights)
series7 = data.tanks.rolling(window=7, center=True).apply(lambda x: np.su

series7[series7.index[0]] = np.sum(data.tanks[:7] * weights[0])
series7[series7.index[1]] = np.sum(data.tanks[1:8] * weights[1])
series7[series7.index[2]] = np.sum(data.tanks[2:9] * weights[2])
series7[series7.index[-3]] = np.sum(data.tanks[-9:-2] * weights[-3])
series7[series7.index[-2]] = np.sum(data.tanks[-8:-1] * weights[-2])
series7[series7.index[-1]] = np.sum(data.tanks[-7:] * weights[-1])

plot = series7.plot(linestyle='-', markevery=100, marker='o', markerfacec

kendall_mean_df = data[['tanks']].join([series3.rename('3'), series5.rena

print(kendall_mean_df)
```

```
[array([ 0.46428571,  0.35714286,  0.25      ,  0.14285714,  0.03571429,
       -0.07142857, -0.17857143]), array([ 0.35714286,  0.28571429,  0.214
28571,  0.14285714,  0.07142857,
        0.        , -0.07142857]), array([0.25      , 0.21428571, 0.178571
43, 0.14285714, 0.10714286,
       0.07142857, 0.03571429]), array([0.14285714, 0.14285714, 0.1428571
4, 0.14285714, 0.14285714,
       0.14285714, 0.14285714]), array([0.03571429, 0.07142857, 0.1071428
6, 0.14285714, 0.17857143,
       0.21428571, 0.25      ]), array([-0.07142857,  0.        ,  0.07142
857,  0.14285714,  0.21428571,
        0.28571429,  0.35714286]), array([-0.17857143, -0.07142857,  0.035
71429,  0.14285714,  0.25      ,
        0.35714286,  0.46428571])]
            tanks          3     5          7
date
2022-02-24    1.0   2.166667   2.4   4.857143
2022-02-25    5.0   2.666667  14.6   8.571429
2022-02-26    2.0  13.000000   8.6  12.928571
2022-02-27   32.0  12.333333  10.6   9.142857
2022-02-28    3.0  15.333333  11.6  12.714286
...           ...        ...   ...        ...
2024-12-26    0.0   0.333333   1.4   1.000000
2024-12-27    1.0   0.333333   0.2   1.857143
2024-12-28    0.0   0.333333   1.4   0.857143
2024-12-29    0.0   2.000000   0.3   0.214286
2024-12-30    6.0   5.000000   3.6   1.857143

[1055 rows x 4 columns]
```



Значення:

Графік демонструє застосування зваженої ковзної середньої для часових рядів із різними ваговими коефіцієнтами. Цей метод забезпечує згладжування даних із врахуванням ваг, які підкреслюють певні значення у вікні (наприклад, центральні чи крайні). Вплив вікон згладжування:

Вікно 3: Чутливе до локальних змін, зберігає більшість коливань. Вікно 5: Більш плавне згладжування, але все ще чутливе до короткотермінових коливань. Вікно 7: Найбільше згладжування, дозволяє чіткіше побачити довгостроковий тренд. Переваги цього підходу:

Метод враховує ваги для покращення оцінки середнього. Підходить для аналізу даних із сильними коливаннями, дозволяючи зосередитися на трендах, а не на шумі. Висновок:

Чітко видно зниження волатильності в даних після згладжування. Найкраще підходить для ситуацій, де необхідно усунути короткострокові аномалії і зосередитися на трендах.

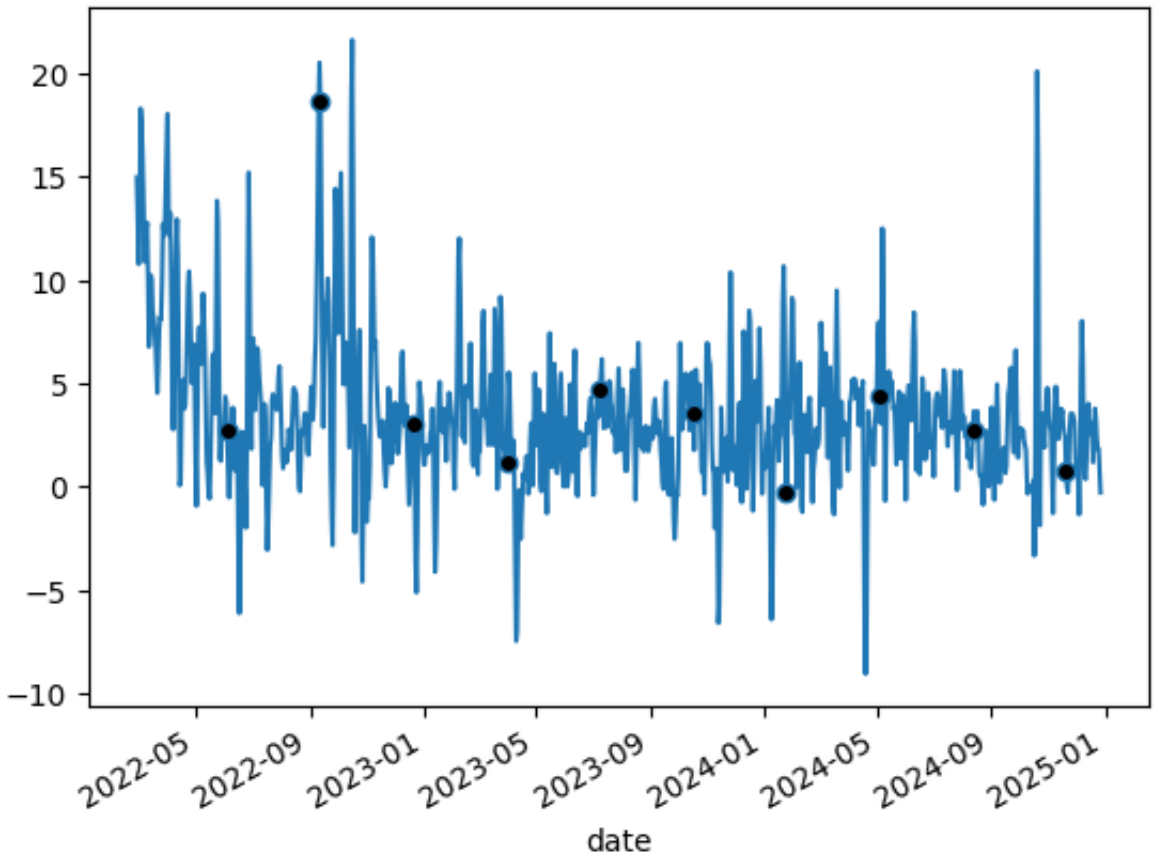
5.3 Метод зваженої ковзної середньої за формулами Полларда

In [21]:
```
# 3
weights = np.array([0.10959, 0.78082, 0.10959])
series3 = data.tanks.rolling(window=3, center=True).apply(lambda x: np.su
```

In [22]:
```
# 5
weights = np.array([-0.07343, 0.293706, 0.559441, 0.293706, -0.07343])
series5 = data.tanks.rolling(window=5, center=True).apply(lambda x: np.su
```

In [23]:
```
# 7
weights = np.array([-0.05874, 0.058741, 0.293706, 0.412587, 0.293706, 0.0
series7 = data.tanks.rolling(window=7, center=True).apply(lambda x: np.su
plot = series7.plot(linestyle='-', markevery=100, marker='o', markerfacec

pollard_mean_df = data[['tanks']].join([series3.rename('3'), series5.rena

print(pollard_mean_df)
```

```
            tanks         3          5          7
date
2022-02-24    1.0       NaN        NaN        NaN
2022-02-25    5.0   4.23287        NaN        NaN
2022-02-26    2.0   5.61647  11.692284        NaN
2022-02-27   32.0  25.53419  18.195762  14.965030
2022-02-28    3.0   7.05483  13.426521  12.751071
...           ...       ...        ...        ...
2024-12-26    0.0   0.10959  -0.146874   0.646152
2024-12-27    1.0   0.78082   0.559441  -0.292293
2024-12-28    0.0   0.10959  -0.146874        NaN
2024-12-29    0.0   0.65754        NaN        NaN
2024-12-30    6.0       NaN        NaN        NaN

[1055 rows x 4 columns]
```



5.5 Медіанна фільтрація

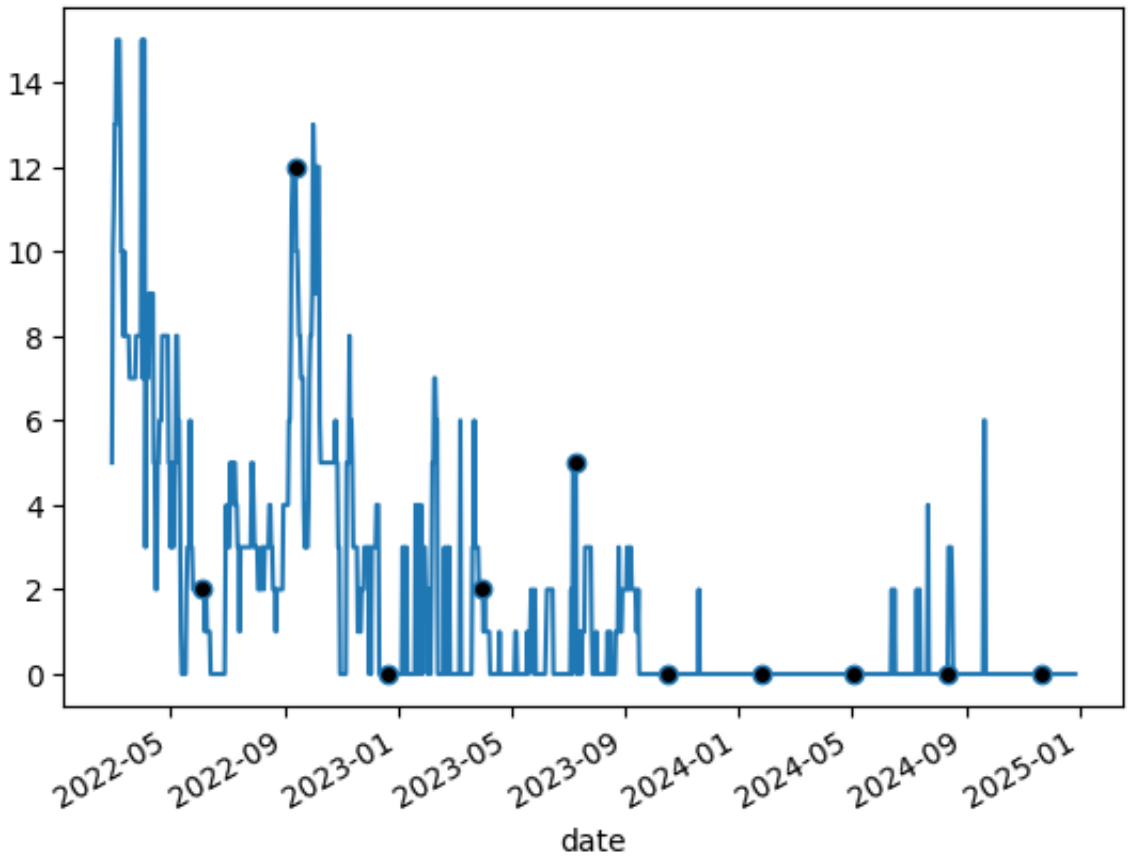
In [24]:
```
# 3
series3 = data.tanks.rolling(window=3, center=True).apply(lambda x: np.me
```

In [25]:
```
# 5
series5 = data.tanks.rolling(window=5, center=True).apply(lambda x: np.me
```

In [26]:
```
# 7
series7 = data.tanks.rolling(window=7, center=True).apply(lambda x: np.me

plot = series7.plot(linestyle='-', markevery=100, marker='o', markerfacec

median_df = data[['tanks']].join([series3.rename('3'), series5.rename('5'
```

```
print(median_df)
```

```
            tanks     3     5     7
date
2022-02-24    1.0   NaN   NaN   NaN
2022-02-25    5.0   2.0   NaN   NaN
2022-02-26    2.0   5.0   3.0   NaN
2022-02-27   32.0   3.0   5.0   5.0
2022-02-28    3.0  11.0  10.0  10.0
...           ...   ...   ...   ...
2024-12-26    0.0   0.0   0.0   0.0
2024-12-27    1.0   0.0   0.0   0.0
2024-12-28    0.0   0.0   0.0   NaN
2024-12-29    0.0   0.0   NaN   NaN
2024-12-30    6.0   NaN   NaN   NaN

[1055 rows x 4 columns]
```



Значення графіка:

Графік демонструє дані після застосування медіанної фільтрації з вікном (window=7). Метод медіанної фільтрації згладжує дані, замінюючи кожне значення вікна на медіану, що дозволяє ефективно усувати викиди.
Особливості:

Видно значне усунення піків (аномальних значень). Графік виглядає "ступінчастим", оскільки медіана створює більш різкі переходи, ніж середнє.
Висновки:

Метод медіанної фільтрації ефективно справляється із шумом та викидами в

даних. Згладжені дані демонструють більш стабільний тренд, хоча частково можуть втрачатися деякі особливості короткострокових коливань.

Різниця між медіанною фільтрацією та ковзним середнім

1. Стійкість до викидів

Ковзне середнє: Чутливе до аномальних сплесків. Якщо у вибірці з'являється велике (або дуже маленьке) значення, середнє суттєво зміщується.

Медіанна фільтрація: Стійка до викидів. Навіть якщо у вікні з'являється екстремальне значення, медіана його "ігнорує" краще, оскільки вона віддзеркалює "центральне" значення.

Висновок: Якщо у даних часті сильні викиди (аномалії), медіанна фільтрація дає стабільніший результат.

2. Тип методу (лінійність / нелінійність)

Ковзне середнє: Лінійний метод: середнє значення завжди можна представити у вигляді зваженої суми вхідних даних.

Медіанна фільтрація: Нелінійний метод: медіана залежить від рангу значень у вікні, тож результат не є лінійною комбінацією вхідних точок.

Висновок: Якщо важлива саме лінійна операція (наприклад, для подальших статистичних або частотних методів), краще ковзне середнє. Якщо треба зберегти тенденції, позбавившись викидів, можна застосовувати медіанну фільтрацію.

3. Форма вихідного сигналу

Ковзне середнє: Згладжує дані більш плавно, утворює "гладкі" переходи, особливо якщо використовується метод зваженого середнього (наприклад, більш високі ваги для центральних значень вікна).

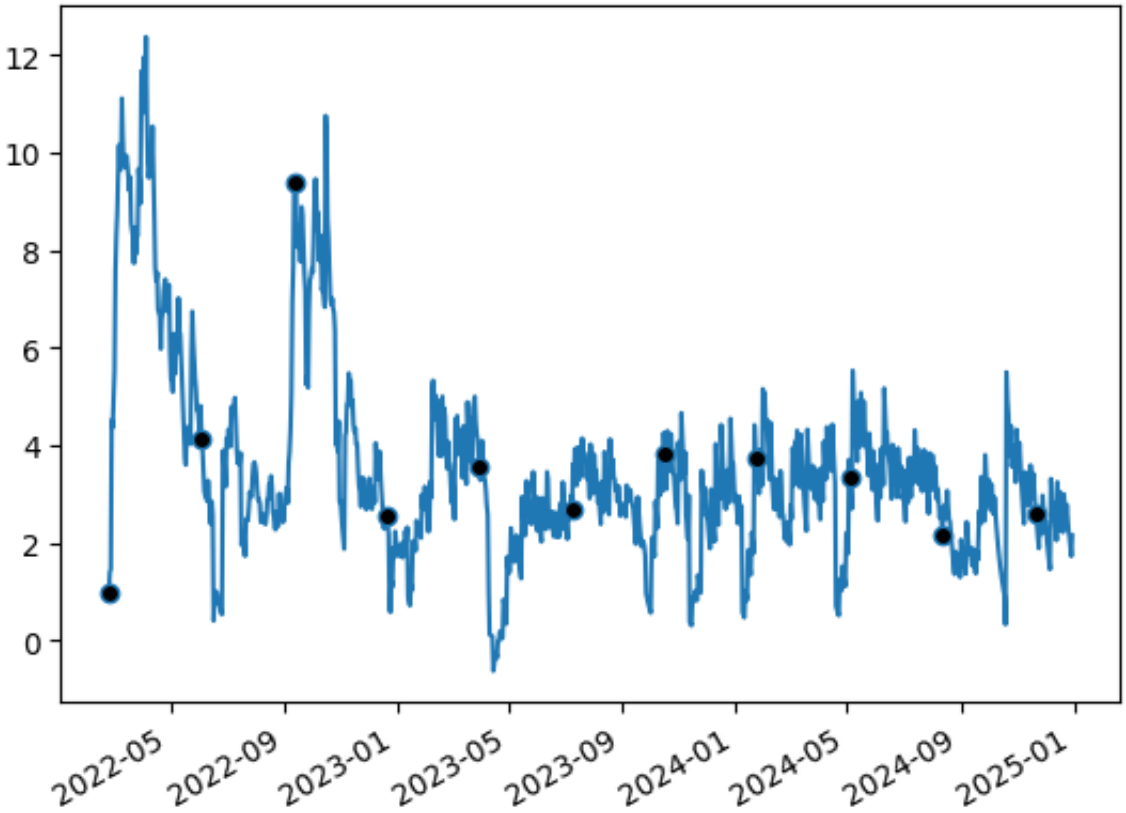
Медіанна фільтрація: Часто утворює "сходинки" (ступінчасті зміни), оскільки кожне вікно повертає одне з фактичних значень, а не середнє між ними. Переходи можуть бути різкіші.

Висновок: Для візуально більш "плавного" ряду краще підходить ковзне середнє. Для видалення різких піків — медіанна фільтрація.

5.6 Експоненційна фільтрація

In [27]:
```
series = pd.Series([])
a = 0.1

items = data.tanks.items()
```

```
index0, value0 = next(items)
series[index0] = value0
prev = value0

for index, value in data.tanks.items():
    prev = value * a + prev * (1 - a)
    series[index] = prev

plot = series.plot(linestyle='-', markevery=100, marker='o', markerfaceco

exp_df = pd.DataFrame({"tanks": data.tanks, "exp": series})
```



Експоненційне згладжування (EWMA):

Графік відображає результати експоненційного згладжування даних. Цей метод використовує параметр згладжування a, який задає вагу нових значень щодо попередніх. У цьому прикладі a = 0.1, тобто новим значенням надається лише 10% ваги, а решта 90% базується на попередніх даних.

Особливості експоненційного згладжування: Сильна інерція: Графік показує плавне слідування за трендом, при цьому локальні коливання сильно приглушуються. Менше чутливе до шуму: Шумові точки згладжуються ефективніше, ніж у методах з фіксованими вікнами.

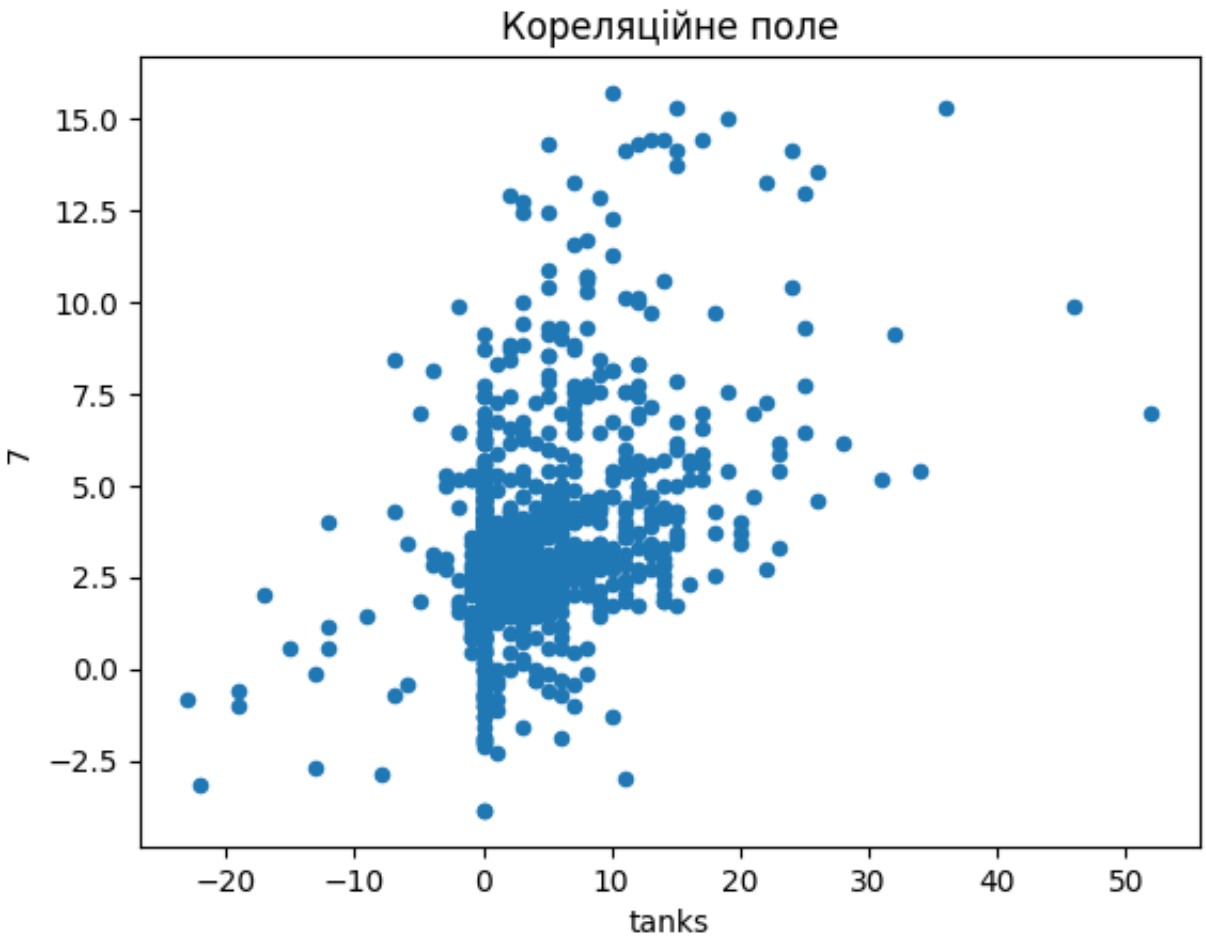
In [28]:
```
# Процедура для аналізу алгоритму згладжування
def analyze_dataframe_series(dataframe):
    columns = list(dataframe.columns)
    column = columns[-1]

    # подати узагальнений графік результатів згладжування для однієї реал
    dataframe[column].plot(linestyle='-', markevery=100, marker='o', mark
```

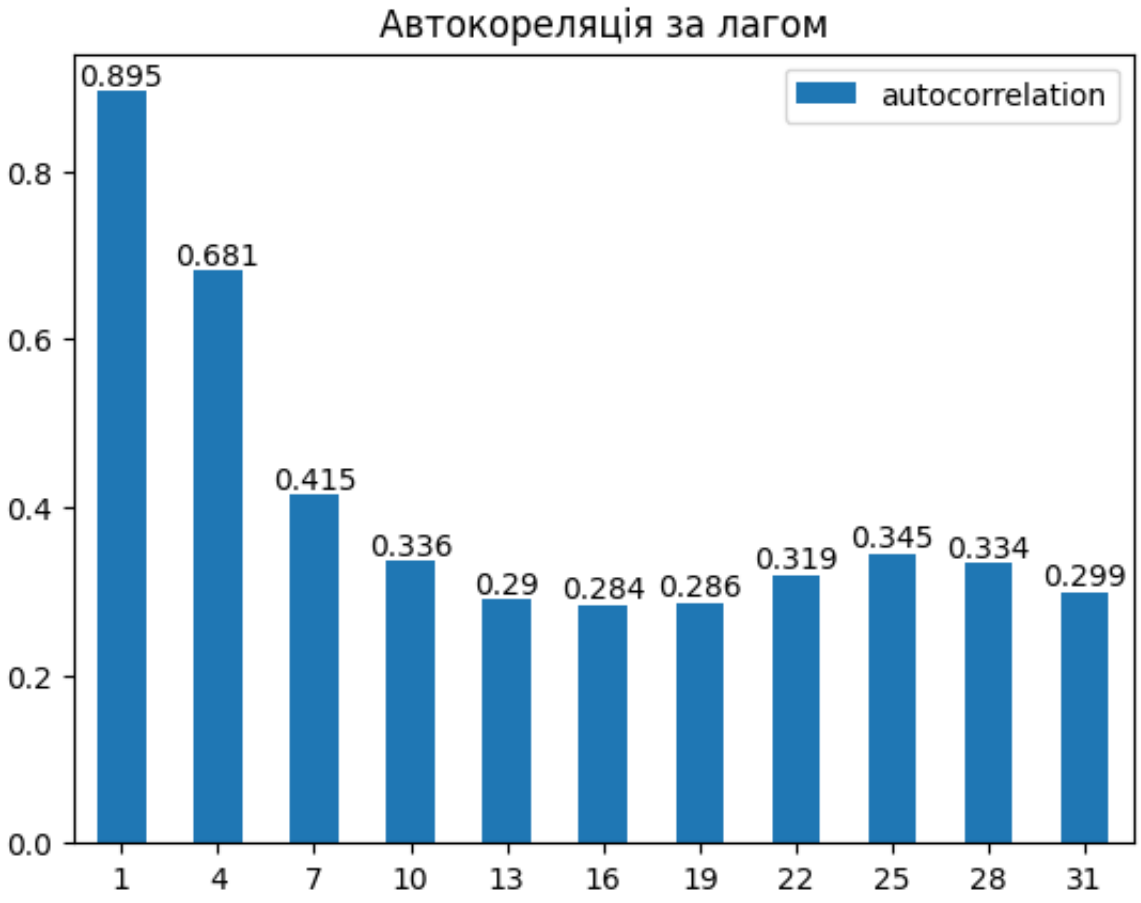
```
    # побудувати кореляційну таблицю для всіх інтервалів згладжування, вк
    print("\nТаблиця кореляції")
    print(dataframe.corr())

    def get_turning_points():
        i1 = dataframe.itertuples(index=False)
        i2 = dataframe.itertuples(index=False)
        i3 = dataframe.itertuples(index=False)
        next(i2)
        next(i3)
        next(i3)

        turning_points = {x: 0 for x in columns}
        for row1, row2, row3 in zip(i1, i2, i3):
            for index in range(len(columns)):
                if np.isnan(row1[index]) or np.isnan(row2[index]) or np.i
                    continue
                if row1[index] > row2[index] < row3[index] or row1[index]
                    turning_points[columns[index]] += 1
        return turning_points

    print("\nПоворотні точки:\n%s" % (get_turning_points()))

    columns = ['Оригінальні дані'] + list(dataframe.columns[1:])

    # Кореляційне поле
    dataframe.plot.scatter(x="tanks", y=column, title="Кореляційне поле")

    # Коефіцієнт кореляції
    print("\nКоеффіціент кореляції між орігінальними даними та згладеними

    # Автокореляція з графіком
    auto_corr = pd.Series([])
    index = pd.Series([])
    for i in range(1, 32, 3):
        auto_corr[i] = round(dataframe[column].autocorr(i), 3)
        index[i] = i
    auto_corr_df = pd.DataFrame({'autocorrelation': auto_corr}, index=ind
    ax = auto_corr_df.plot.bar(rot=0, title="Автокореляція за лагом")

    for container in ax.containers:
        ax.bar_label(container)

    # Розбити одну з послідовностей на три рівні частини.
    part_size = dataframe[column].count() // 3 - 2
    s1 = dataframe[column][3:part_size + 3].rename('part1').reset_index(d
    s2 = dataframe[column][part_size + 3: 2 * part_size + 3].rename('part
    s3 = dataframe[column][2 * part_size + 3: 3 * part_size + 3].rename('

    # Побудувати для них кореляційну матрицю.
    corr_df = pd.DataFrame({'part1': s1, 'part2': s2, 'part3': s3})
    print("\nКореляція між частинами виборки зі згладеними даними (коефіц
    print(corr_df.corr())
```

6. Аналіз зваженої ковзної середньої за формулами Кендела

In [29]:
```
analyze_dataframe_series(kendall_mean_df)
```

```
Таблиця кореляції
          tanks         3         5         7
tanks  1.000000  0.515273  0.395968  0.446411
3      0.515273  1.000000  0.787794  0.748593
5      0.395968  0.787794  1.000000  0.863492
7      0.446411  0.748593  0.863492  1.000000

Повортні точки:
{'tanks': 509, '3': 307, '5': 360, '7': 355}

Коеффіцієнт кореляції між орігінальними даними та згладеними
0.4464114303198823

Кореляція між частинами виборки зі згладеними даними (коефіцієнти множинної кореляції)
          part1     part2     part3
part1  1.000000  0.031052 -0.143623
part2  0.031052  1.000000  0.097094
part3 -0.143623  0.097094  1.000000
```





Кореляційна матриця

	tanks	3	5	7

	tanks	3	5	7
tanks	1.000000	0.515273	0.395968	0.446411
3	0.515273	1.000000	0.787794	0.748593
5	0.395968	0.787794	1.000000	0.863492
7	0.446411	0.748593	0.863492	1.000000

Взаємозв'язок з оригінальними даними: Найвищий коефіцієнт кореляції з оригінальними даними (0.515) має ряд із вікном 3. Для більших вікон (5 і 7) кореляція дещо знижується: 0.396 і 0.446 відповідно. Це свідчить, що вікна 5 і 7 сильніше "згладжують" коливання, тому відхиляються від оригінального ряду.

Взаємозв'язок між згладженими рядами: Ряди з більшими вікнами (5 і 7) мають вищу кореляцію між собою (0.863), ніж із вікном 3. Це означає, що чим більший розмір вікна, тим подібніші між собою згладжені ряди (адже вони згладжують ще "сильніше").

Поворотні точки {'tanks': 509, '3': 307, '5': 360, '7': 355}

Оригінальний ряд має 509 локальних екстремумів (максимуми/мінімуми). Згладжені ряди (3, 5, 7) мають меншу кількість екстремумів: 307, 360 і 355 відповідно. Це підтверджує, що згладжування "згладжує" й екстремуми. Чим менше вікно (3), тим сильніше воно відслідковує коливання, але все ж зменшує кількість поворотних точок порівняно з оригіналом.

Коефіцієнт кореляції з оригінальними даними 0.446411

Це збігається з рядком із вікном 7, бо саме його ми аналізували останнім. Значення ~0.45 свідчить про помірну кореляцію: зростання/спад оригінального ряду частково віддзеркалюється у згладженому.

Кореляція між частинами вибірки

	part1	part2	part3
part1	1.000000	0.031052	−0.143623
part2	0.031052	1.000000	0.097094
part3	−0.143623	0.097094	1.000000

Низькі та переважно близькі до нуля кореляції між різними частинами означають, що в різних періодах ряду (після згладження) немає сильної лінійної залежності. Тобто тренди в першій, другій та третій частинах відрізняються, і їх поведінка не надто схожа одна на одну.

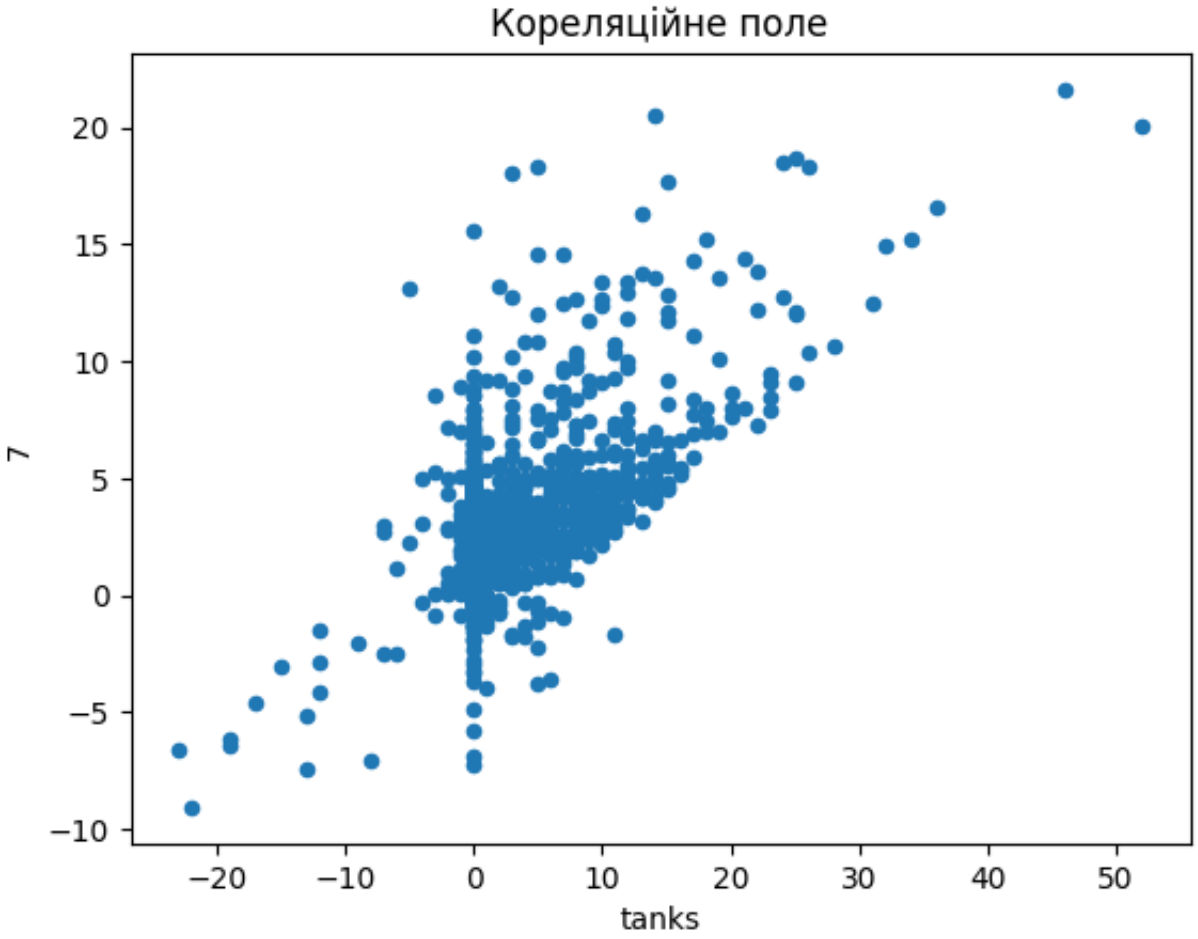
Підсумок Вікно 3 краще зберігає коливання, має найбільшу кореляцію з оригіналом. Вікна 5 та 7 призводять до сильнішого згладжування, зменшують волатильність і кількість екстремумів, проте й віддаляються від початкових даних. Усі згладжені варіанти (3, 5, 7) суттєво знижують кількість локальних максимумів і мінімумів, демонструючи, що зважена ковзна середня дійсно "вирівнює" дані. Низька кореляція між частинами (part1, part2, part3) натякає, що в різні періоди часового ряду картина змінювалася, і єдиний тренд не

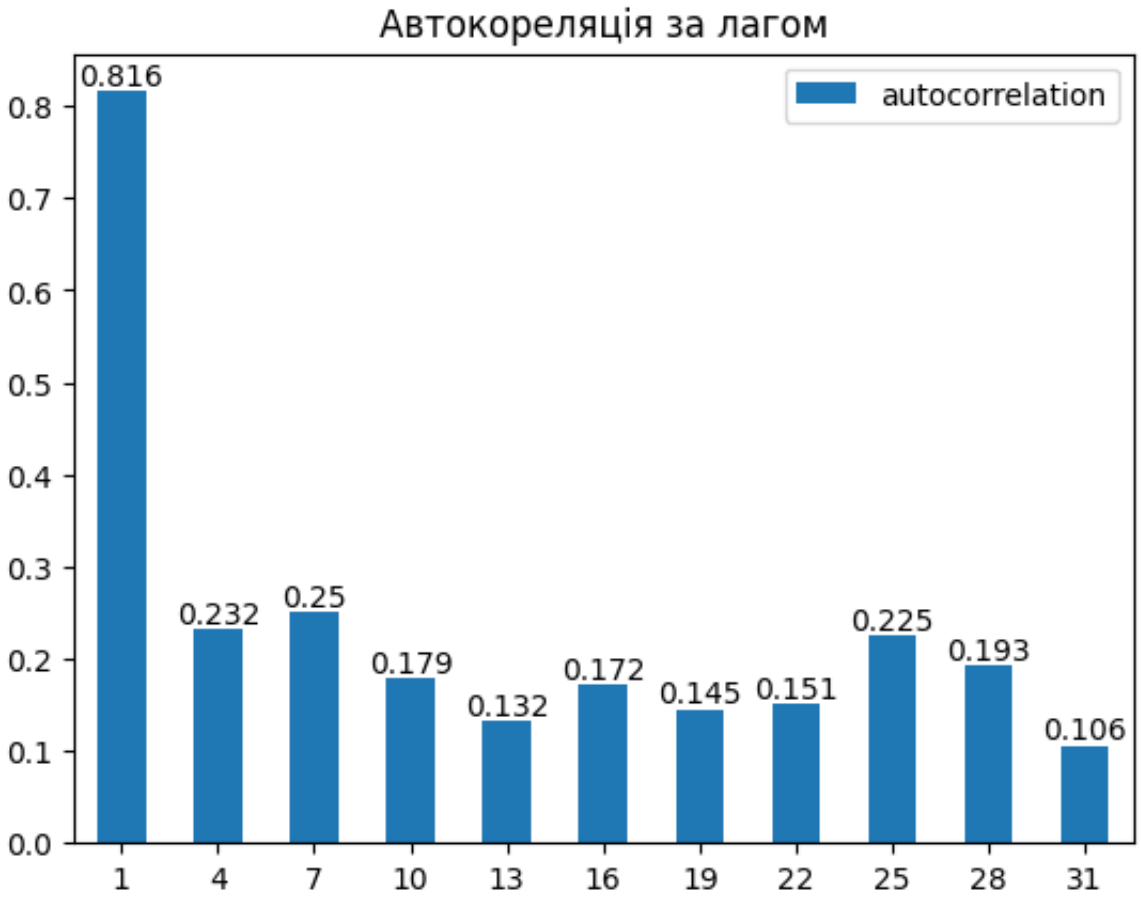
домінує по всій історії.

7. Згладжування за формулами з Полларда

In [30]:
```
analyze_dataframe_series(pollard_mean_df)
```

```
Таблиця кореляції
          tanks         3         5         7
tanks  1.000000  0.979588  0.773425  0.627664
3      0.979588  1.000000  0.880956  0.764274
5      0.773425  0.880956  1.000000  0.917124
7      0.627664  0.764274  0.917124  1.000000

Повортні точки:
{'tanks': 509, '3': 620, '5': 542, '7': 397}

Коеффіцієнт кореляції між орігінальними даними та згладеними
0.6276638563872977

Кореляція між частинами виборки зі згладеними даними (коефіцієнти множинної кореляції)
          part1     part2     part3
part1  1.000000 -0.011414 -0.104728
part2 -0.011414  1.000000 -0.021470
part3 -0.104728 -0.021470  1.000000
```





1. Кореляційна матриця

tanks	3	5	7

```
        tanks  1.000000  0.979588  0.773425  0.627664
        3      0.979588  1.000000  0.880956  0.764274
        5      0.773425  0.880956  1.000000  0.917124
        7      0.627664  0.764274  0.917124  1.000000
```

Висока кореляція ряду tanks із вікном 3 (≈0.98) Це свідчить, що згладження з найменшим вікном (3) дуже схоже на оригінальний ряд і мало змінює його “структуру”.

Помірно висока кореляція з вікном 5 (≈0.77) і з вікном 7 (≈0.63). Зі зростанням розміру вікна коефіцієнт кореляції з оригінальними даними знижується, оскільки згладжування стає сильнішим.

Взаємна кореляція між згладженими рядами: 3 ↔ 5 : 0.88 5 ↔ 7 : 0.92 3 ↔ 7 : 0.76 Чим більші вікна (5 і 7), тим більша схожість між собою (0.92), бо вони більше “вирівнюють” коливання.

2. Поворотні точки

{'tanks': 509, '3': 620, '5': 542, '7': 397}

Оригінальний ряд має 509 екстремумів (локальних максимумів і мінімумів).

Вікно 3 дає 620 поворотних точок, що навіть більше, ніж у первинних даних.

Це може свідчити про “перегини” у формулі згладжування, коли ряд стає трохи “нерівнішим” у деяких ділянках (залежно від ваг коефіцієнтів Полларда).

Вікно 5 – 542 екстремуми: дещо згладжується, але все ще коливання зберігаються.

Вікно 7 – 397 екстремумів: найменше поворотних точок, найбільший ступінь згладження.

3. Коефіцієнт кореляції між оригінальним рядом та згладженим (вікно 7)

0.6276638563872977

Це відповідає рядку з вікном 7, який ми аналізували “останнім” у порівнянні з оригінальним. Значення ~0.63 свідчить про помірно сильний зв’язок між “найбільш згладженим” рядом і оригінальним, хоч і нижчий, ніж з меншими вікнами.

4. Кореляція між частинами вибірки

```
              part1     part2     part3
        part1  1.000000 -0.011414 -0.104728
        part2 -0.011414  1.000000 -0.021470
        part3 -0.104728 -0.021470  1.000000
```

Низькі (і від’ємні) кореляції між різними періодами даних (після згладження).

Це свідчить про відсутність стійких повторюваних трендів між різними частинами часової вибірки — тобто дані в кожному відрізку можуть змінюватися за власним “сценарієм”. Загальний підсумок

Найменше вікно (3) дає найвищу схожість із оригіналом (кореляція 0.98), але водночас може збільшувати кількість “псевдо”-екстремумів.

Середнє вікно (5) зберігає компроміс: достатньо згладжує коливання (зменшуючи поворотні точки порівняно з вікном 3) і при цьому кореляція з оригіналом (0.77) досить висока.

Найбільше вікно (7) найсильніше згладжує (найменше поворотних точок), проте віддаляється від оригіналу (кореляція 0.63).

Різні частини ряду (part1, part2, part3) лишаються малопов’язаними одна з одною (кореляції біля нуля), що підкреслює варіативність поведінки даних на різних часових інтервалах. Таким чином, згладжування за формулами Полларда дозволяє вибирати міру згладження (через розмір вікна), однак слід враховувати можливий “перепад” у кількості екстремумів при найменшому вікні.

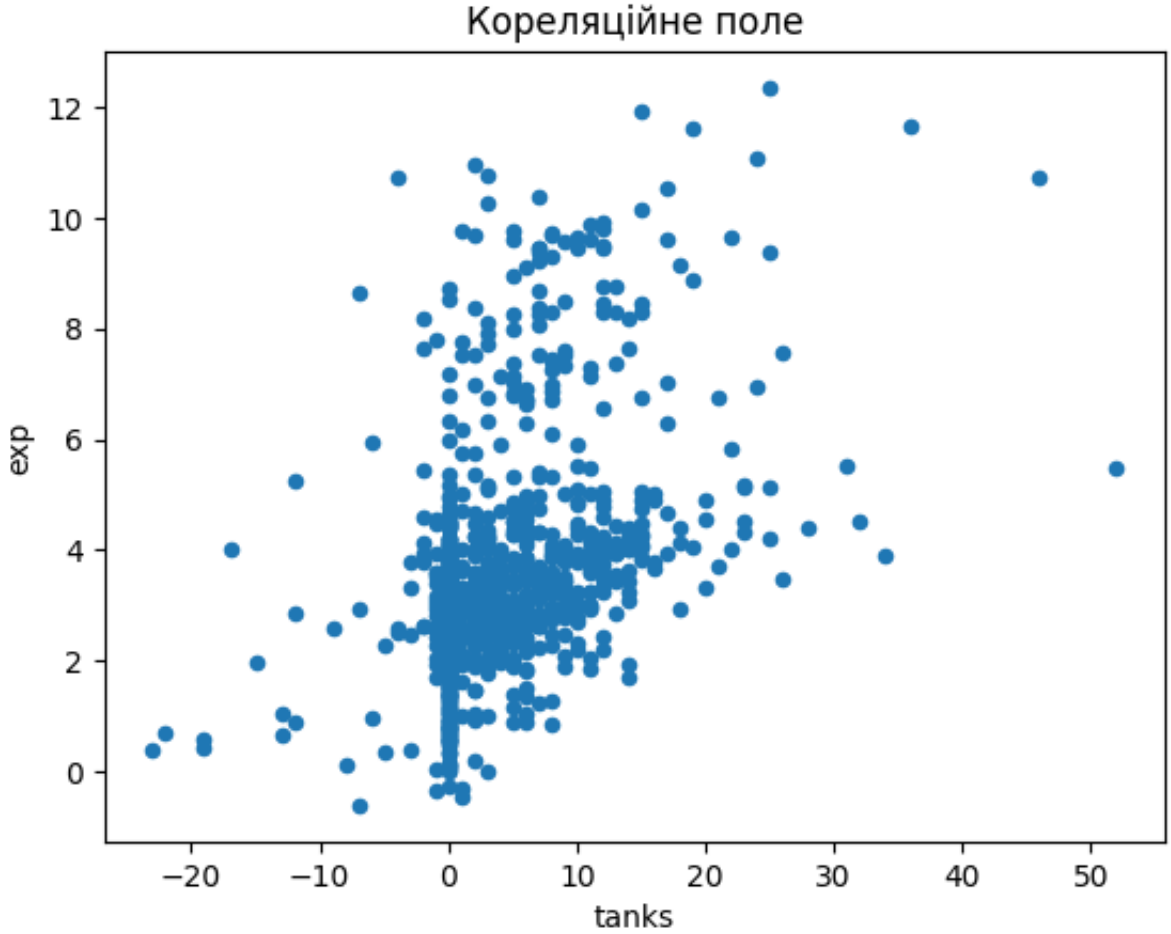
Якщо мета — сильно зменшити волатильність, можна використовувати більше вікно (7), і навпаки, якщо потрібно зберегти максимальну схожість з оригіналом — краще брати менше вікно (3).

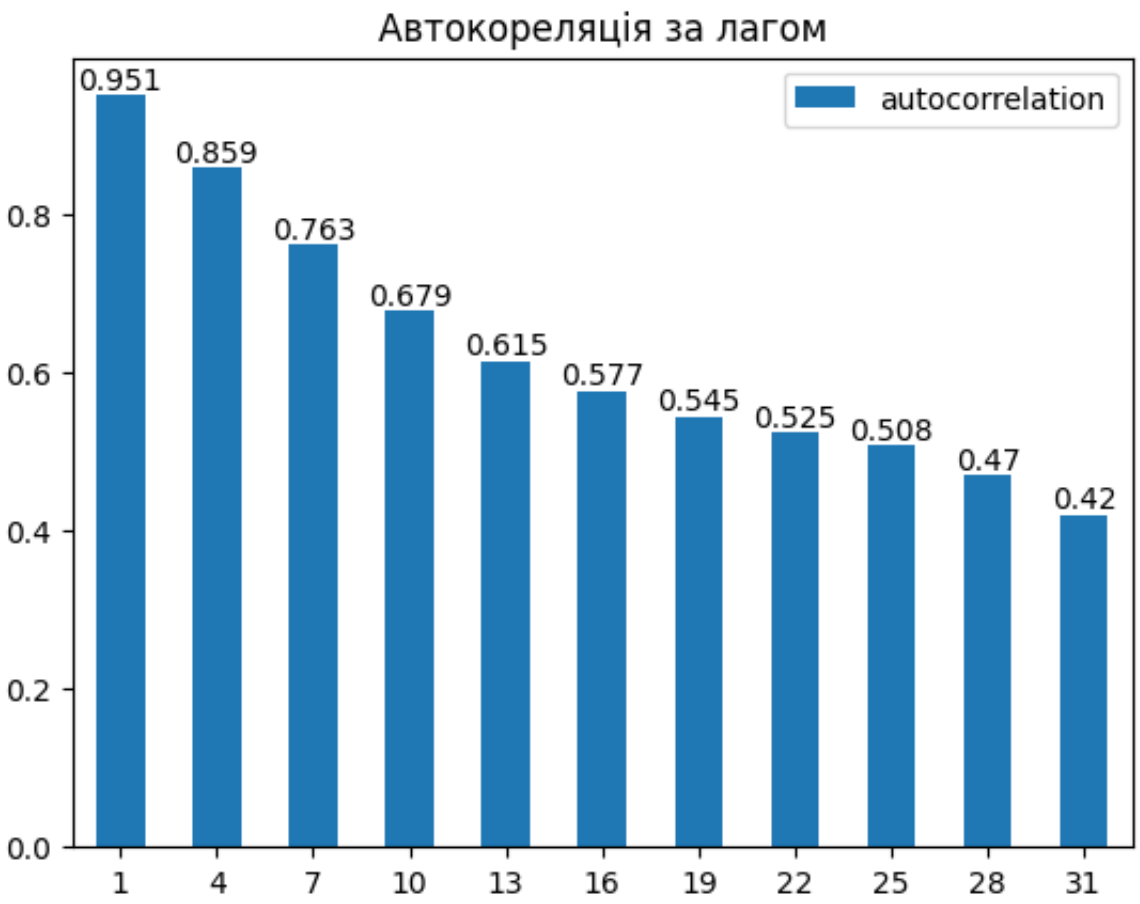
8. Експоненціальне згладжування

In [31]:
```
analyze_dataframe_series(exp_df)
```

```
Таблиця кореляції
          tanks       exp
tanks  1.000000  0.458824
exp    0.458824  1.000000

Повортні точки:
{'tanks': 509, 'exp': 563}

Коеффіцієнт кореляції між орігінальними даними та згладеними
0.4588239906461523

Кореляція між частинами виборки зі згладеними даними (коефіцієнти множинно
ї кореляції)
          part1     part2     part3
part1  1.000000  0.169263 -0.090846
part2  0.169263  1.000000  0.021137
part3 -0.090846  0.021137  1.000000
```


Графік результатів згладжування
12
10
8
6
4
2
0
2022-05
2022-09
2023-01
2023-05
2023-09
2024-01
2024-05
2024-09
2025-01


Кореляційне поле
exp
12
10
8
6
4
2
0
−20
−10
0
10
20
30
40
50
tanks



1. Кореляційна матриця

	tanks	exp
tanks	1.000000	0.458824
exp	0.458824	1.000000

Експоненційне згладжування з невисоким коефіцієнтом a розмазує дані, тому короткочасні піки та “шум” досить сильно згладжуються.

2. Поворотні точки

{'tanks': 509, 'exp': 563} Оригінальний ряд: 509 локальних екстремумів. Експоненційно згладжений: 563 локальних екстремумів.

3. Коефіцієнт кореляції між оригінальними даними та згладженими

0.4588239906461523 Значення ~0.46 вказує на помірну позитивну лінійну залежність.

Лінійна залежність - з збільшенням одного значення - збільшується інше значення

4. Кореляція між частинами вибірки

	part1	part2	part3
part1	1.000000	0.169263	-0.090846
part2	0.169263	1.000000	0.021137

```
part3 -0.090846  0.021137  1.000000
```

Кореляції між різними відрізками (частинами) згладженої вибірки досить низькі (близькі до нуля, навіть негативні).

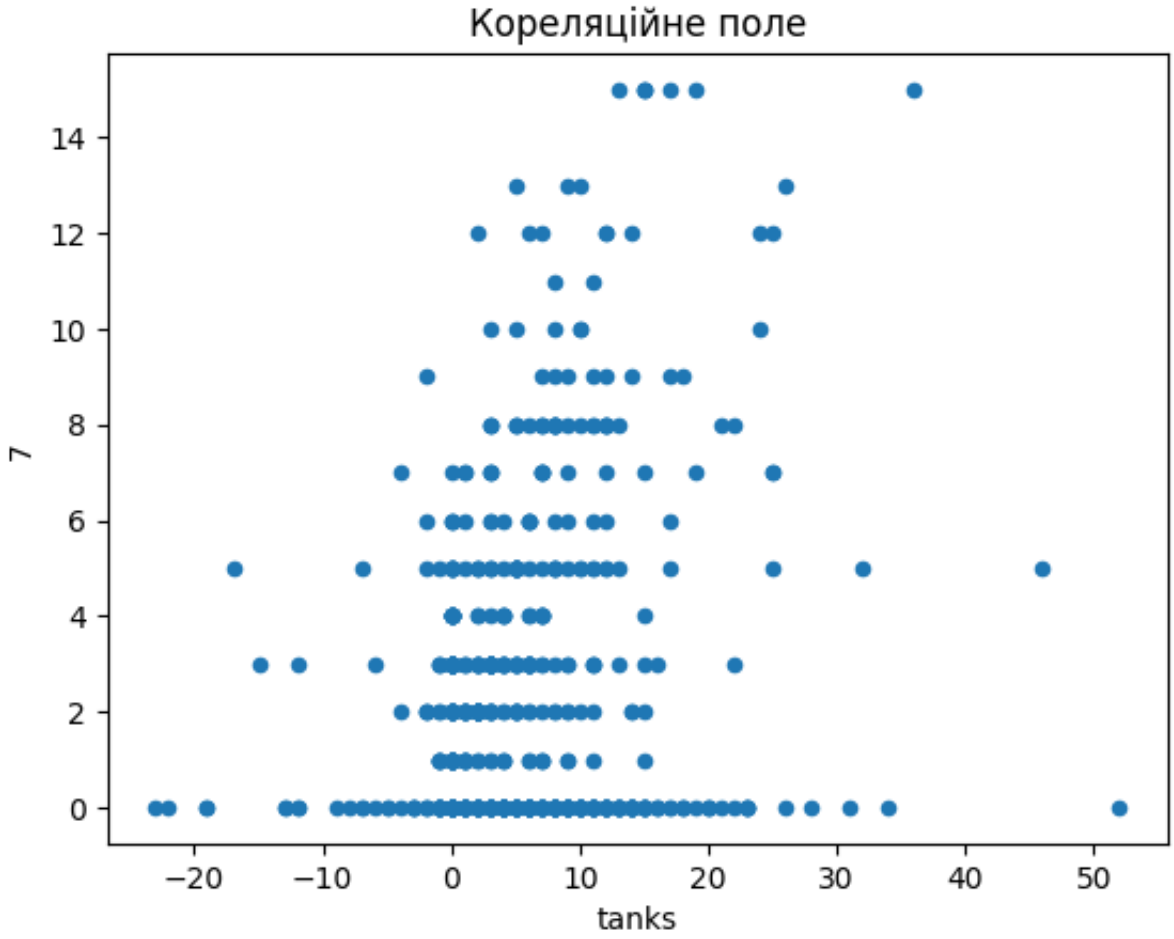
Це означає, що тренд у кожній частині ряду може відрізнятися, і немає сильної повторюваності патерну з однієї частини до іншої.

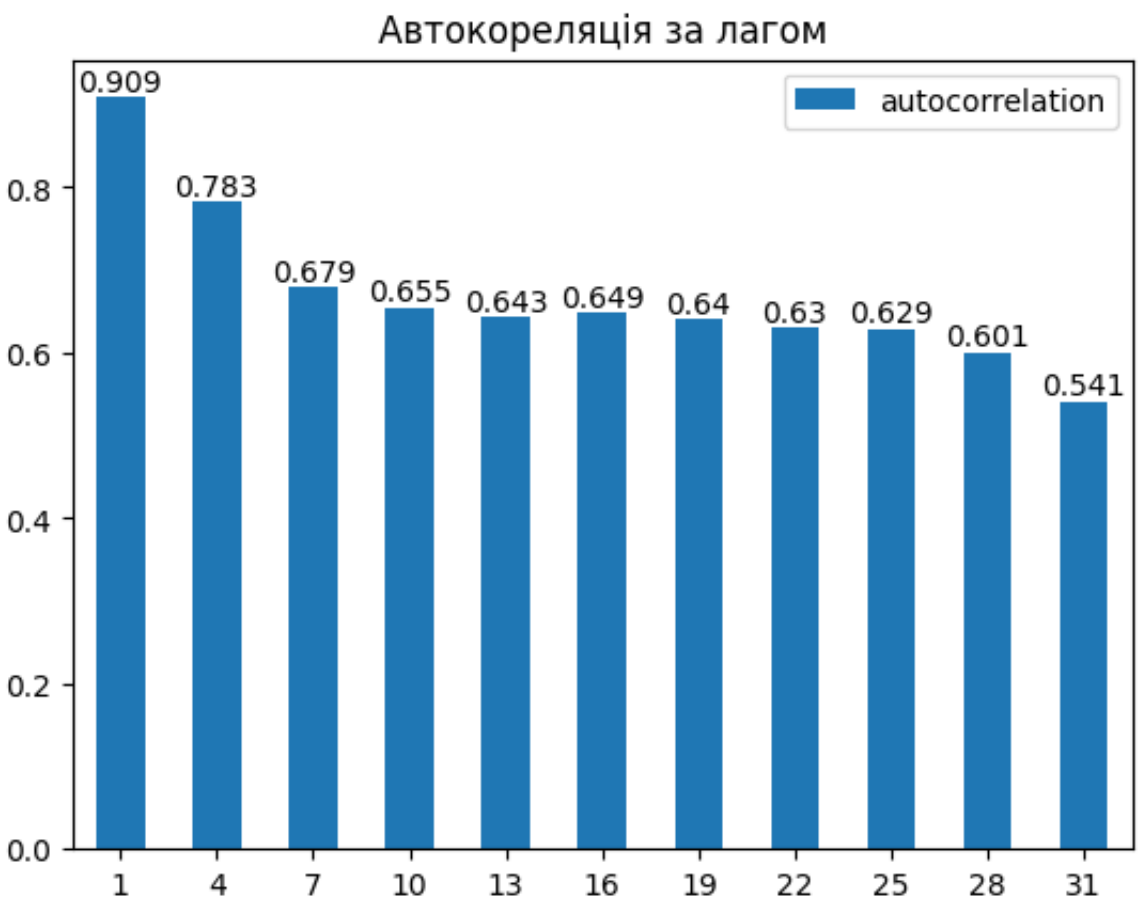
9. Медіанне згладжування

In [32]:
```
analyze_dataframe_series(median_df)
```

```
Таблиця кореляції
          tanks         3         5         7
tanks  1.000000  0.303908  0.332265  0.342628
3      0.303908  1.000000  0.789582  0.787351
5      0.332265  0.789582  1.000000  0.892020
7      0.342628  0.787351  0.892020  1.000000

Повортні точки:
{'tanks': 509, '3': 138, '5': 108, '7': 75}

Коеффіцієнт кореляції між орігінальними даними та згладеними
0.3426284254041169

Кореляція між частинами виборки зі згладеними даними (коефіцієнти множинної кореляції)
          part1     part2     part3
part1  1.000000  0.217383 -0.004364
part2  0.217383  1.000000  0.055510
part3 -0.004364  0.055510  1.000000
```


Графік результатів згладжування
14
12
10
8
6
4
2
0
2022-05
2022-09
2023-01
2023-05
2023-09
2024-01
2024-05
2024-09
2025-01
date


Кореляційне поле
7
14
12
10
8
6
4
2
0
−20
−10
0
10
20
30
40
50
tanks



Загальний висновок

Низька кореляція з оригіналом вказує на те, що медіанне згладжування радикально “зрізає” коливання та викиди.

Суттєве зменшення поворотних точок (з 509 до 75–138) показує сильне приглушення змін у ряді. Великі вікна (5, 7) формують досить подібні ряди між собою (висока взаємна кореляція), але віддаляються від оригіналу більше, ніж менше вікно (3).

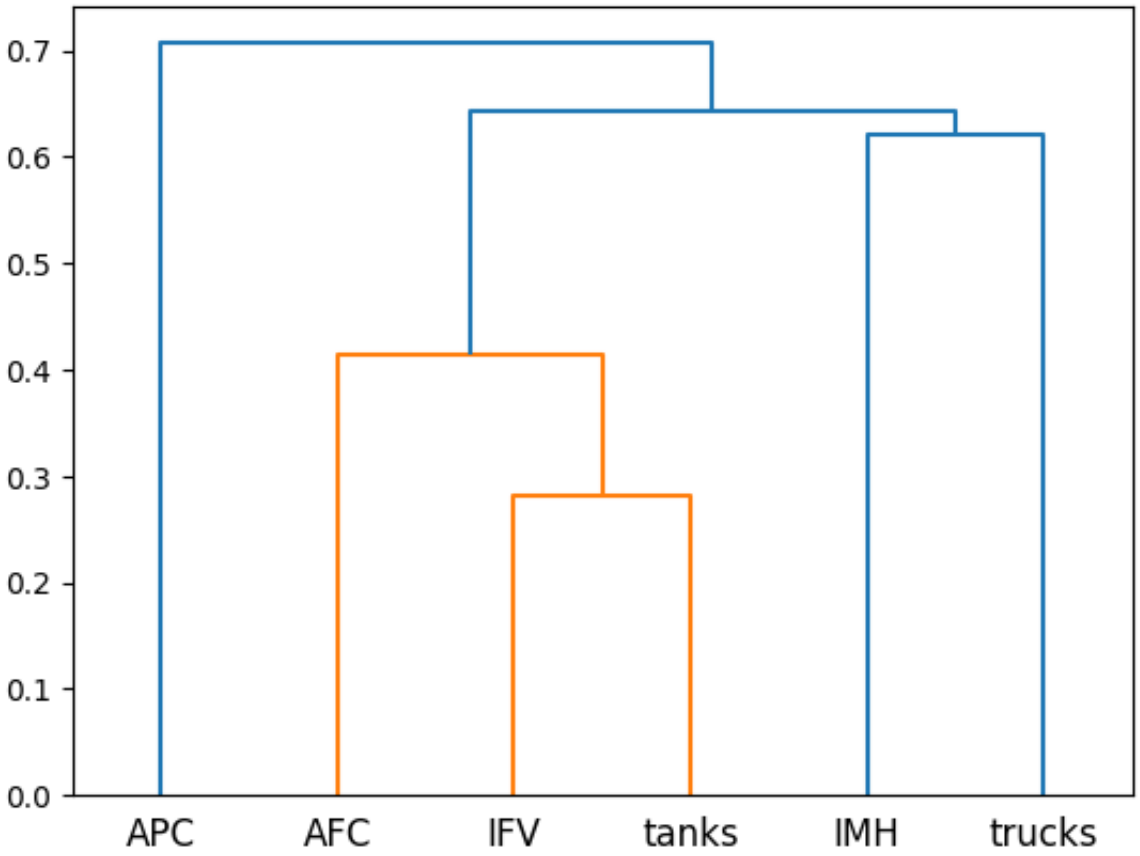
Різні частини ряду продовжують слабо корелювати між собою, підтверджуючи, що структура даних відрізняється по періодах.

Загалом медіанне згладжування є дуже ефективним для видалення викидів, але при цьому може суттєво змінити оригінальний ряд, зменшуючи кореляцію з ним і “стискаючи” динаміку до менш мінливої кривої.

10. Ієрархічний агломеративний кластерний аналіз багатомірних даних

In [33]:
```
from scipy.cluster import hierarchy as hc
from scipy.spatial.distance import pdist

# зважена відстань на основі кореляції. Використовуємо зважену тому що да
corr = 1 - data.corr()
```

```
corr_condensed = hc.distance.squareform(corr) # convert to condensed

# використовуємо стратегію групового середнього
z = hc.linkage(corr_condensed, method='average')
relabel = {
    'armoured fighting vehicles': 'AFC',
    'armoured personnel carriers': 'APC',
    'infantry fighting vehicles': 'IFV',
    'infantry mobility vehicles': 'IMH',
    'tanks': 'tanks',
    'trucks, vehicles and jeeps': 'trucks'
}

labels = [relabel[item] for item in list(corr.columns)]

dendrogram = hc.dendrogram(z, labels=labels)
plt.show()
```



Висновки: Близькі зв'язки: Змінні, які з'єднуються на малих висотах, мають сильну кореляцію. Наприклад, якщо AFC та IFV з'єднуються раніше за інші, це може свідчити, що ці два параметри часто мають схожу поведінку в даних.

Великі відстані: Якщо змінна (наприклад, tanks) з'єднується з іншими лише на великій висоті, це може вказувати на її відносно незалежну природу або слабкий зв'язок з іншими.

Підіб'ємо підсумки:

Початковий огляд та гістограма

Часові ряди:

Первинний графік (лінійний часовий ряд) показав високу волатильність у перші місяці (кінець зими – весна 2022-го), з періодичними піками далі. До 2024 року кількість знищеної техніки коливалася, але середні значення знизилися порівняно з початком повномасштабної війни. Гістограма:

Розподіл здебільшого зміщений у бік значень від 0 до 10 (найбільша частка спостережень). Присутні і викиди: трапляються поодинокі великі значення (до 50+), та траплялися від'ємні записи (можливо, наслідок помилок в даних або специфічних обрахунків). Кумулятивна сума:

Лінія кумулятивної суми має майже рівномірно зростаючий характер, що свідчить про постійну інтенсивність знищення техніки. Серйозних "провалів" чи "стрибків" (з погляду накопиченого підсумку) не було.

Ковзні середні (у тому числі зважені)

Прості/зважені ковзні середні з вікнами 3, 5, 7, 15:

Чим більше вікно, тим сильніше згладжування: від 3 (де короткострокова волатильність іще помітна) до 15 (де графік стає максимально плавним). Спостерігалася тенденція до зменшення коливань у часі, проте збільшувалася різниця з оригінальними даними.

Формули Кендела:

У порівнянні з оригіналом, найбільша кореляція була з меншим вікном (3); зі збільшенням вікна (5, 7) ряд віддалявся від оригінального. Кількість поворотних точок зменшувалася (але, залежно від ваг, десь могла зростати), вказуючи на загальне згладження "піків".

Формули Полларда:

Для вікна 3 отримували високу кореляцію з оригінальними даними, але місцями навіть збільшувалась кількість локальних екстремумів. Вікна 5, 7 давали менший зв'язок з оригіналом, однак суттєво знижували волатильність. Загалом, ковзні середні (прості чи зважені) чітко показали, що дані мають аномалії та нерівномірний тренд із численними піками, особливо на початку. Після згладження видно відносну стабілізацію до 2023–2024 рр.

Медіанне згладжування

Медіана ефективно "зрізає" викиди, тому в результаті кореляція з оригіналом часто була низькою (близько 0.3–0.34), адже з ряду фактично вилучаються різкі перепади. Суттєво зменшується кількість поворотних точок: з 500+ до 75–138 залежно від розміру вікна. Найкраще підходить у випадках, де важливо усунути аномалії, проте треба враховувати, що частково "втрачається" тренд і знижується схожість з оригіналом.

Експоненціальне згладжування

Використовувалося зі швидкістю згладжування a = 0.1 Кореляція з оригіналом виявилася на рівні ~0.46, що свідчить про помірний зв'язок. Поворотних точок навіть стало більше (563 проти 509), що вказує на "інерційну" природу методу: на деяких інтервалах експоненційне згладжування формувало додаткові "хвилі" при наздоганянні різких змін.

Ієрархічний кластерний аналіз

Використано кореляційну відстань (1 – corr) для порівняння різних типів техніки: armoured fighting vehicles (AFC), armoured personnel carriers (APC), infantry fighting vehicles (IFV), infantry mobility vehicles (IMH), tanks, trucks, vehicles and jeeps (trucks).

Дендрограма із "average" linkage показала, як ці змінні групуються за схожістю: Якщо, наприклад, AFC і APC з'єднуються на низькому рівні, це означає високу кореляцію між темпами знищень. "Віддаленіша" змінна свідчить про відсутність тісних зв'язків з іншими. Цей метод дозволяє кластеризувати типи техніки за їх поведінкою (наприклад, якщо якісь категорії знищувалися приблизно однаковими темпами чи в аналогічні періоди).

Загальні тенденції та підсумки

Волатильний початок (2022 р.):

Значні пікові значення (15+), що свідчить про інтенсивні бої й масовані втрати техніки на старті.

Стабілізація / зниження коливань:

Ближче до 2023–2024 рр. середні значення стають нижчими, а кількість "приблизно нульових" днів зростає, проте трапляються поодинокі "сплески". Розподіл (за гістограмою): Переважання малих (0–10) значень, натяк на відносно часту невелику кількість знищень на день (але з періодичними різкими сплесками).

Згладжування:

Показало, що коротке вікно (3, 5) зберігає більше деталей та екстремумів; Велике вікно (7, 15) або сильно "рівняє" ряд, сильно знижуючи волатильність. Медіанні фільтри найрізкіше прибирають викиди, але можуть сильно спотворити криву. Експоненційне згладжування з малим a дає додаткову інерцію в ряді й може збільшувати локальні екстремуми.

Кластерний аналіз:

Дозволяє виявити, які категорії техніки (танки, броньовані машини, вантажівки тощо) підпадають під схожі патерни знищення (за часом чи обсягами). Потенційно корисний для подальшого стратегічного аналізу, щоб зрозуміти, чи

існує “супутнє знищення” певних типів техніки в однакові періоди.

Ключові висновки

Основна динаміка знищень була найбільш інтенсивною в перші місяці вторгнення, надалі коливання знизилися (але не зникли повністю).

Кількість аномальних піків суттєво зменшується після медіанного або сильного ковзного згладжування; отже, викиди (до 50+ одиниць техніки за раз) відіграють значну роль у первинній статистиці.

Використання різних методів згладження (ковзне середнє, медіанне, експоненційне) дало змогу побачити різні “ракурси” поведінки ряду: Від більш детального (з багатьма екстремумами) до більш усередненого (де проглядається лише загальний тренд). Кластери за кореляцією між різними типами техніки відображають їх схожі патерни знищення — це може вказати на тактику, умови боїв або пріоритети враження конкретних цілей.

Низька кореляція між частинами вибірки (part1, part2, part3) свідчить про те, що за весь період 2022–2024 рр. поведінка знищень техніки змінювалася, й повторювані закономірності в різні періоди були слабо виражені.

Отже

Загалом, дослідження дає комплексне уявлення про динаміку знищення російської техніки протягом російсько-української війни (2022–2024). Застосовані графіки, фільтри та кластерний аналіз показали багатогранність даних: наявність високих піків, аномалій, зниження інтенсивності в окремі періоди, а також різний ступінь подібності між типами знищеної техніки. Отримані результати можна використовувати для подальшого прогнозування, моделювання логістики, а також для оцінювання бойових тактик та стратегій.