In [1]:

```
# generate oryx data based on scraped data from https://github.com/leedrake5
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 Comb
                    continue
                if row['equipment_type'].startswith('Losses excluding Recon
                    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 Communicatior

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

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

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

                # calculate difference
                today_losses[row['equipment_type']] = last_data[row['equipme
                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 carr

    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

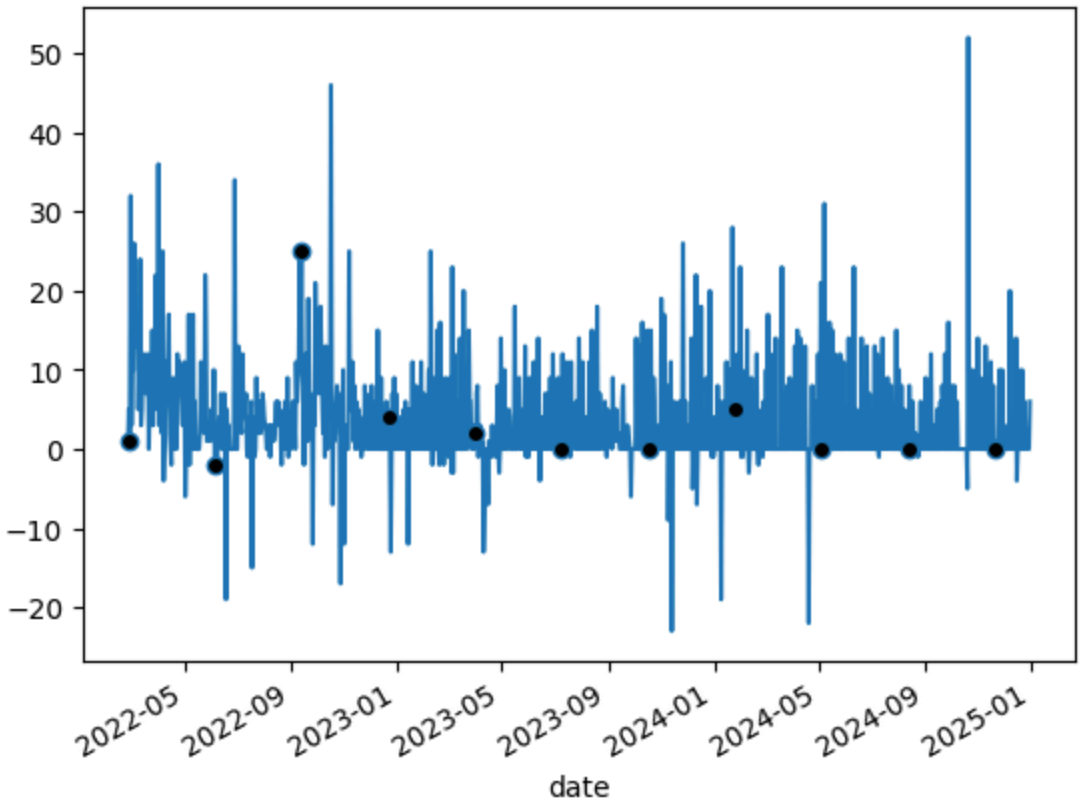
In [5]:
```
data.info()
```

```
<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

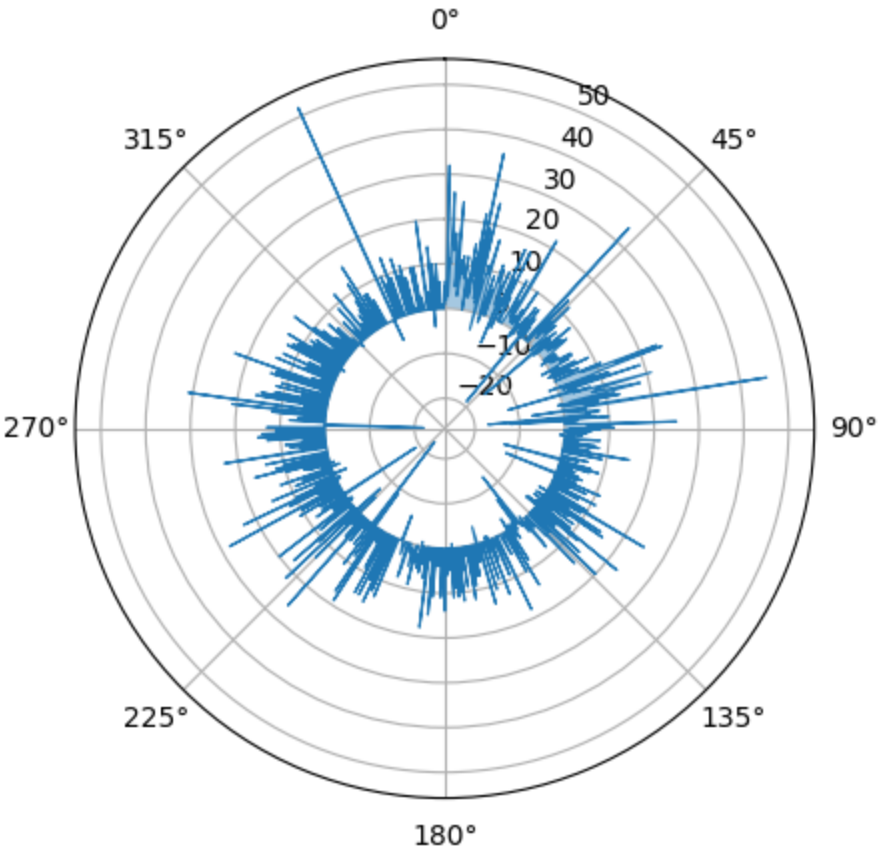
In [6]:
```
data.set_index('date', inplace=True)
plot = data['tanks'].plot(linestyle='-', markevery=100, marker='o', markerfa
```



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.

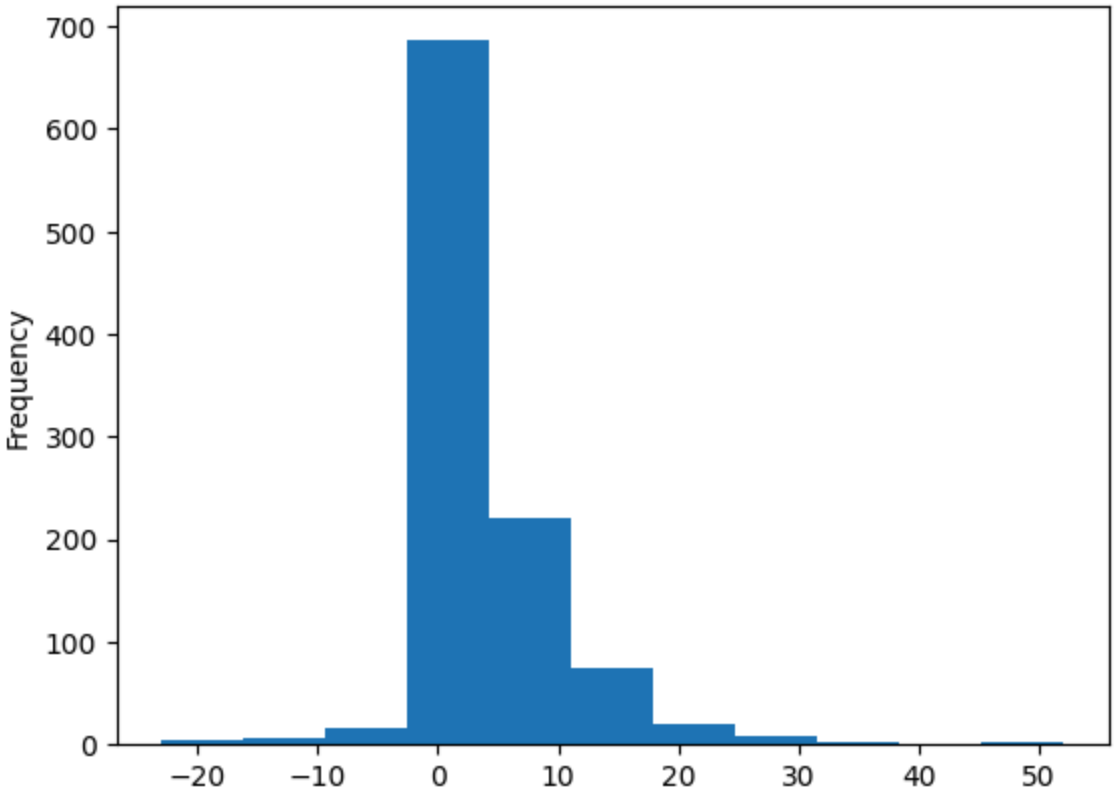
In [9]:
```
print("Результати описової статистики")
print("Показники/Значення")
print("Середнє: %s" % data.tanks.mean())
print("Стандартна помилка: %s" % (data.tanks.std() / data.tanks.count() ** 0
print("Медіана: %s" % data.tanks.median())
print("Мода: %s" % data.tanks.mode()[0])
print("Стандартне відхилення: %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

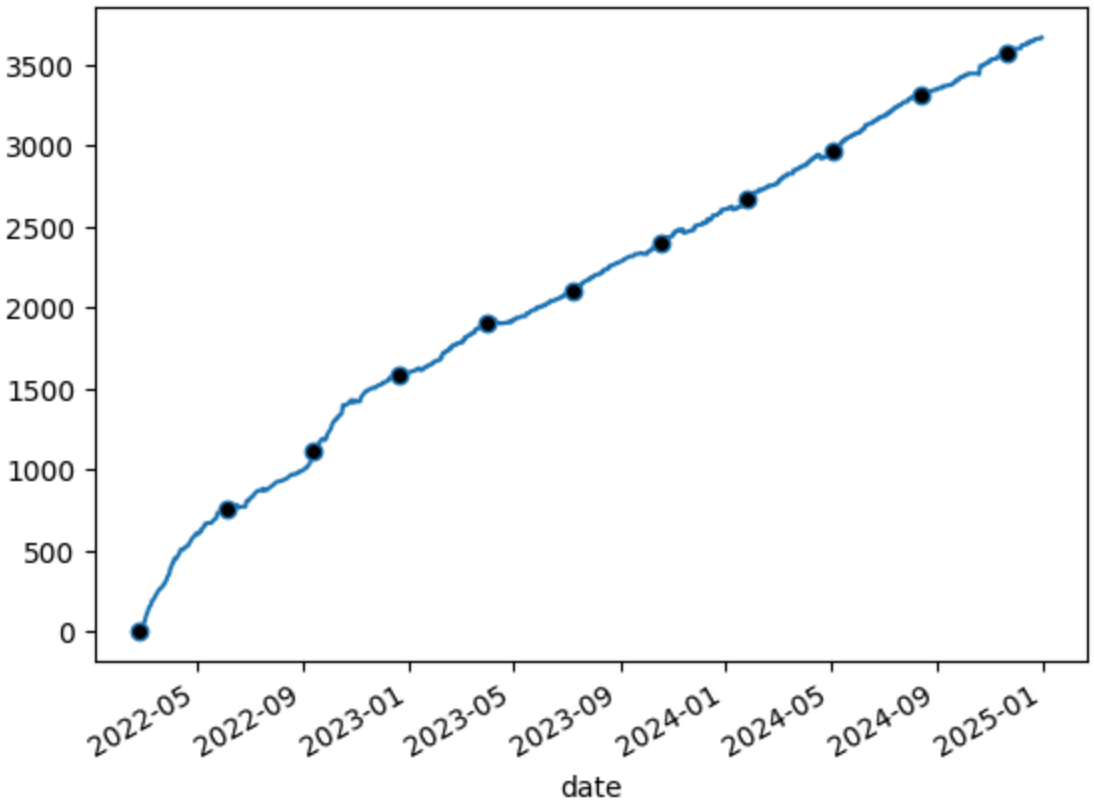
In [10]:
```
# формула Стерджеса
k = int(1 + np.log2(data.shape[0]))
```

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

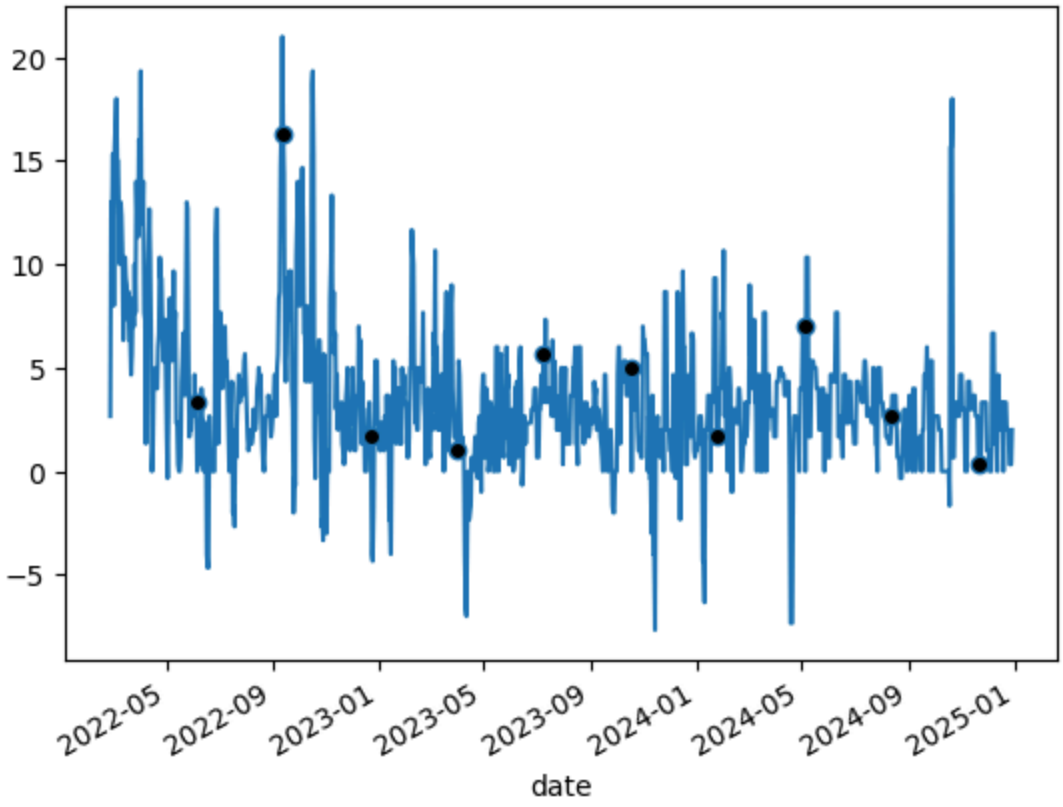


4.2.2. Побудова кумуляти.

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



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



5. Виявлення тенденції часового ряду методами згладжування

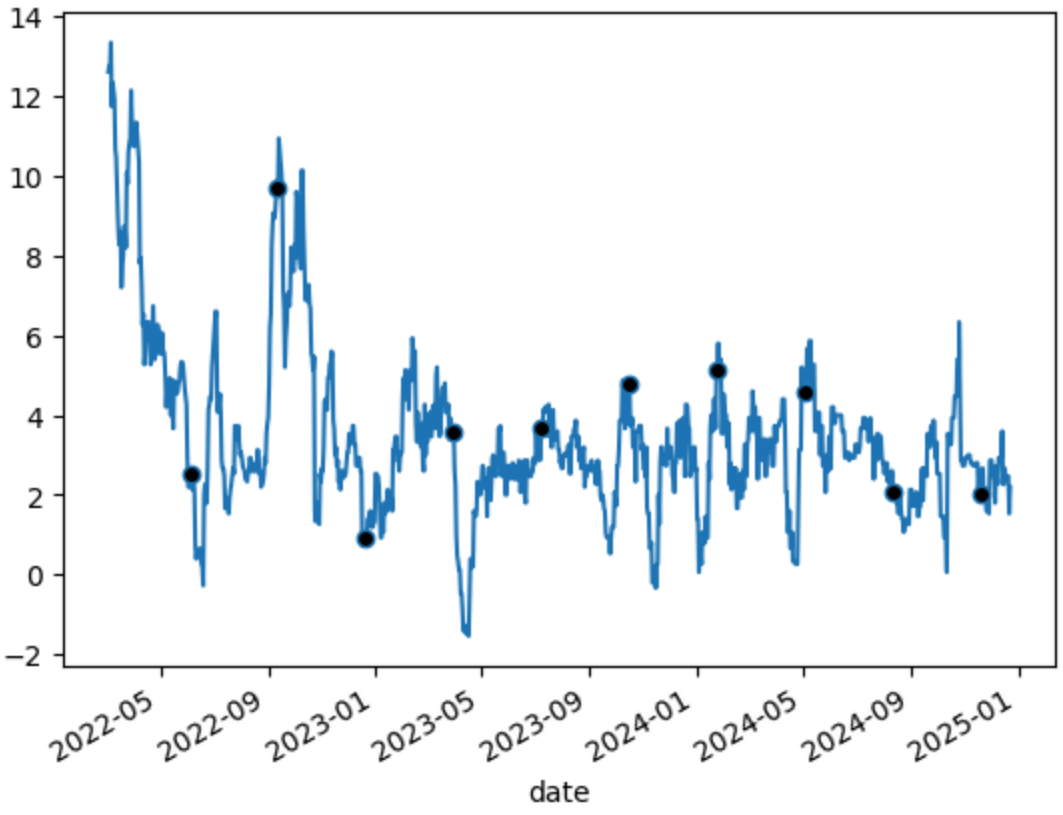
5.2 Метод ковзної середньої

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='-',
```



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.sum(w

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, 0.333333
33, 0.33333333]), array([-0.16666667,  0.33333333,  0.83333333])]
```

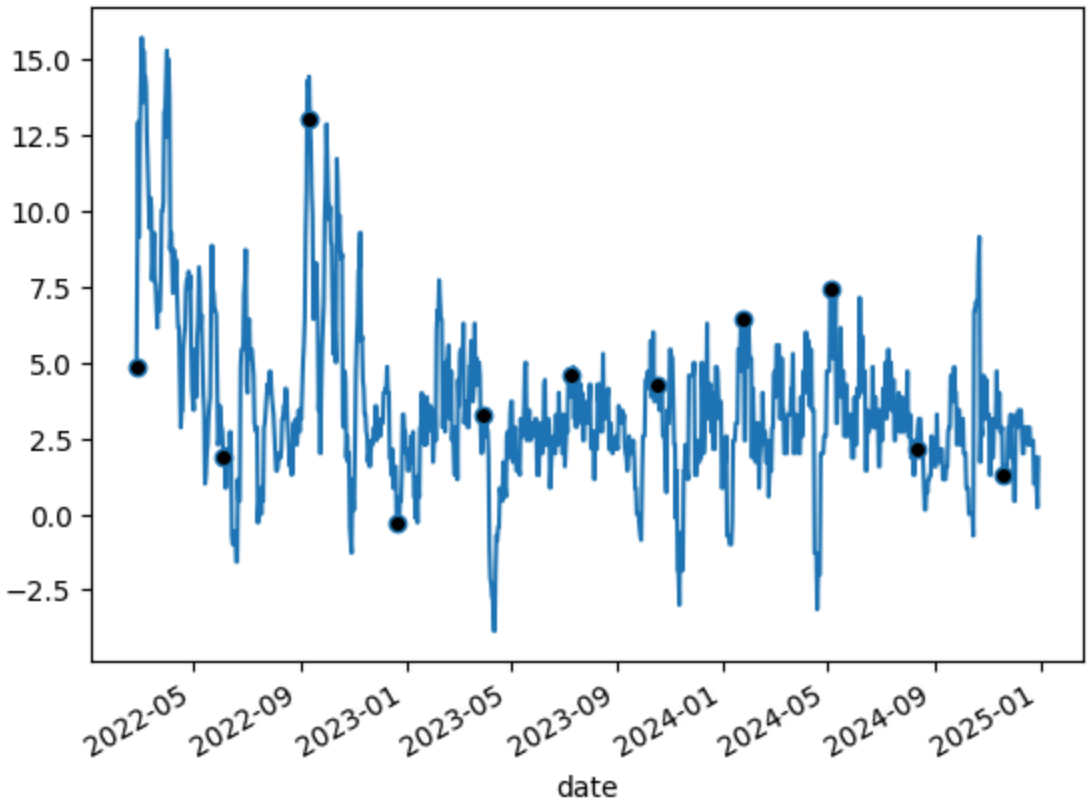
In [19]:

```
# 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.sum(w

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]), ar
ray([0.2, 0.2, 0.2, 0.2, 0.2]), array([0.1, 0.2, 0.3, 0.4, 0.5]), array([-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.sum(w

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', markerfacecolo

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

print(kendall_mean_df)
```

```
[array([ 0.46428571,  0.35714286,  0.25      ,  0.14285714,  0.03571429,
       -0.07142857, -0.17857143]), array([ 0.35714286,  0.28571429,  0.21428
571,  0.14285714,  0.07142857,
        0.        , -0.07142857]), array([0.25      , 0.21428571, 0.1785714
3, 0.14285714, 0.10714286,
       0.07142857, 0.03571429]), array([0.14285714, 0.14285714, 0.14285714,
0.14285714, 0.14285714,
       0.14285714, 0.14285714]), array([0.03571429, 0.07142857, 0.10714286,
0.14285714, 0.17857143,
       0.21428571, 0.25      ]), array([-0.07142857,  0.        ,  0.0714285
7,  0.14285714,  0.21428571,
        0.28571429,  0.35714286]), array([-0.17857143, -0.07142857,  0.03571
429,  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]
```



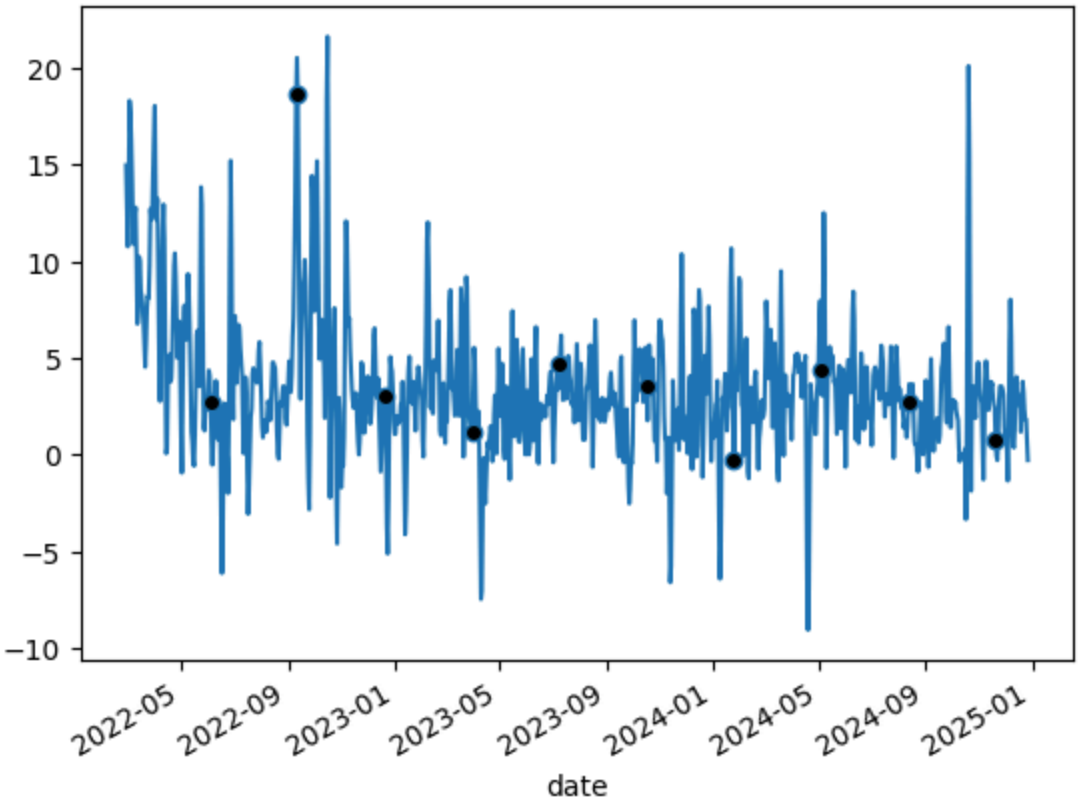
5.3 Метод зваженої ковзної середньої за формулами Полларда

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

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.sum(w
```

In [23]:
```
# 7
weights = np.array([-0.05874, 0.058741, 0.293706, 0.412587, 0.293706, 0.0587
series7 = data.tanks.rolling(window=7, center=True).apply(lambda x: np.sum(w
plot = series7.plot(linestyle='-', markevery=100, marker='o', markerfacecolo

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

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

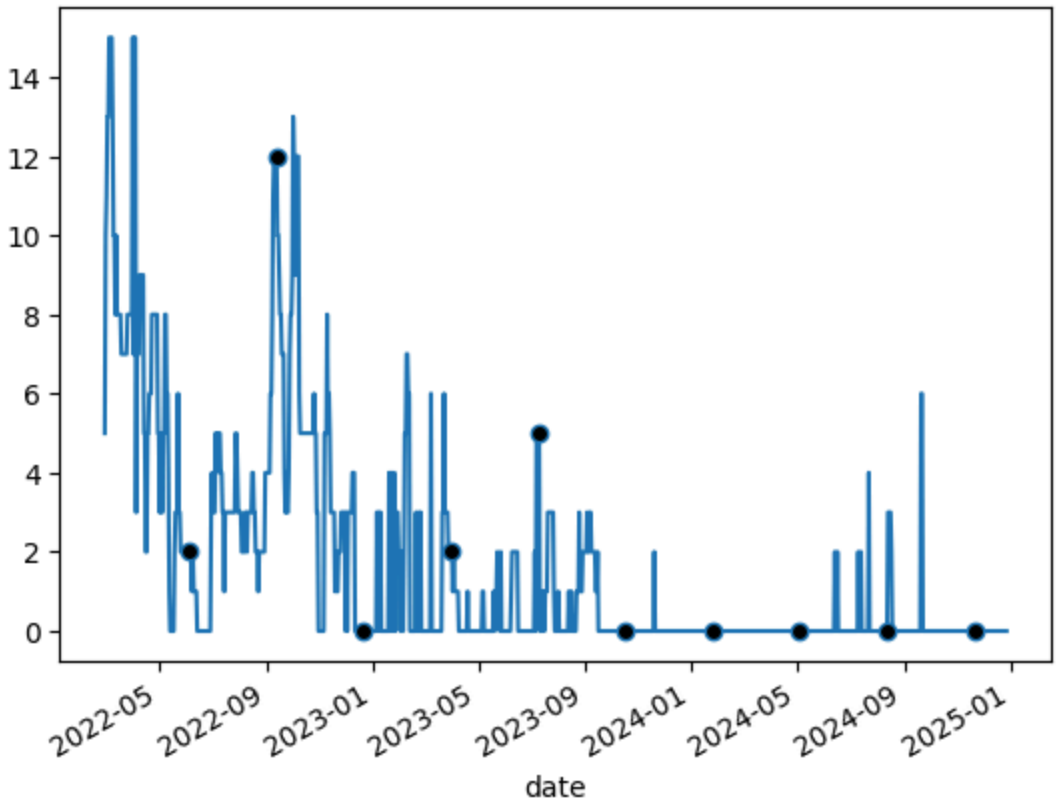
In [24]:
```
# 3
series3 = data.tanks.rolling(window=3, center=True).apply(lambda x: np.media
```

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

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

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

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

print(median_df)
```

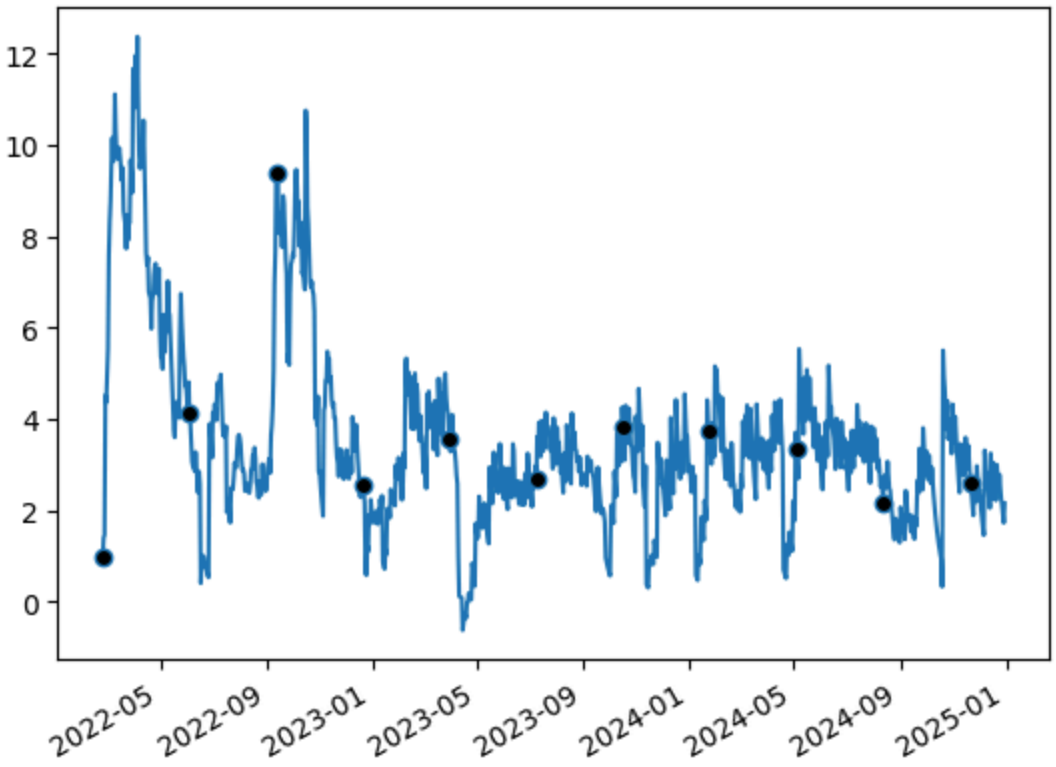
```
            tanks     3     5     7
date
2022-02-24    1.0   NaN   NaN   NaN
2022-02-25    5.0   2.0   NaN   NaN
2022-02-26    2.0   5.0   3.0   NaN
2022-02-27   32.0   3.0   5.0   5.0
2022-02-28    3.0  11.0  10.0  10.0
...           ...   ...   ...   ...
2024-12-26    0.0   0.0   0.0   0.0
2024-12-27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



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', markerfacecolor

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



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 реаліза
    dataframe[column].plot(linestyle='-', markevery=100, marker='o', markerf

    # побудувати кореляційну таблицю для всіх інтервалів згладжування, включ
    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.isna
```

```
                    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\n%

    # 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=index)
    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(drop
    s2 = dataframe[column][part_size + 3: 2 * part_size + 3].rename('part2')
    s3 = dataframe[column][2 * part_size + 3: 3 * part_size + 3].rename('par

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

6. Аналіз зваженої ковзної середньої за формулами Кендела

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

Таблиця кореляції

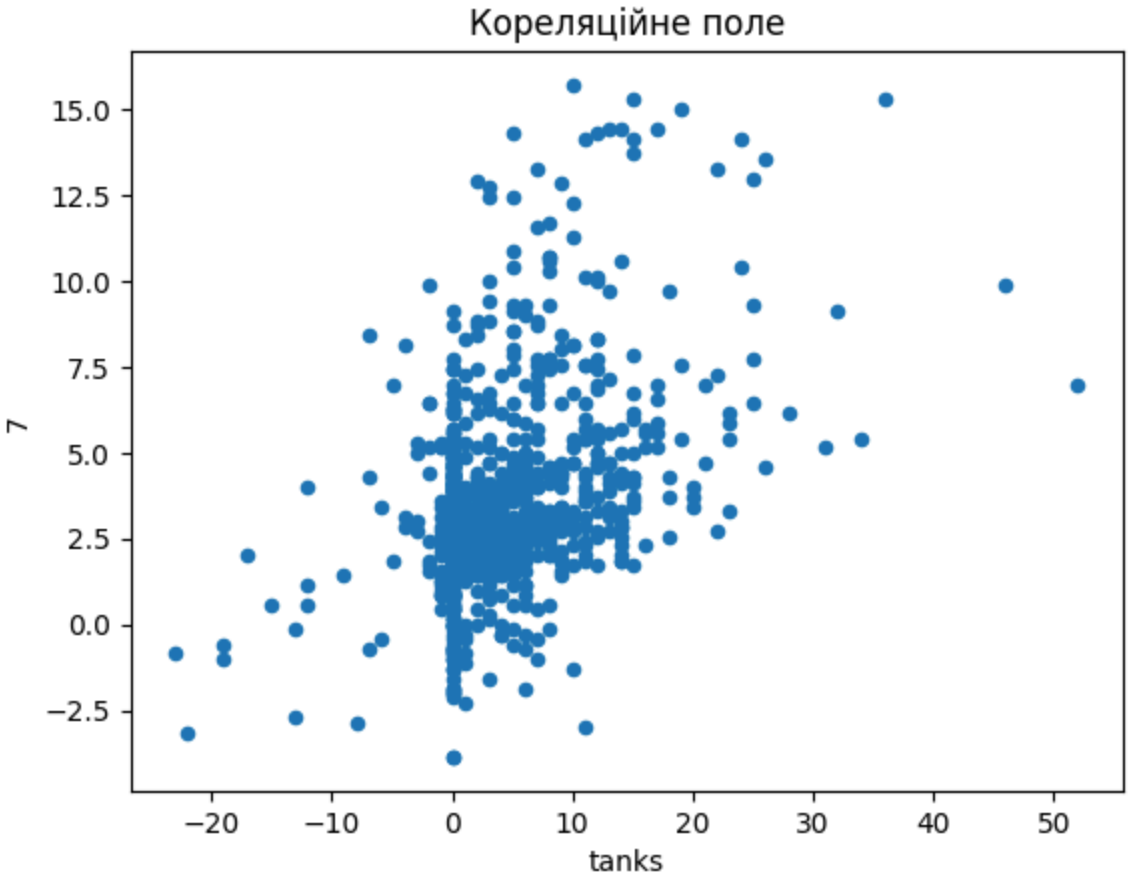
	tanks	3	5	7
tanks	1.000000	0.515273	0.395968	0.446411
3	0.515273	1.000000	0.787794	0.748593
5	0.395968	0.787794	1.000000	0.863492
7	0.446411	0.748593	0.863492	1.000000

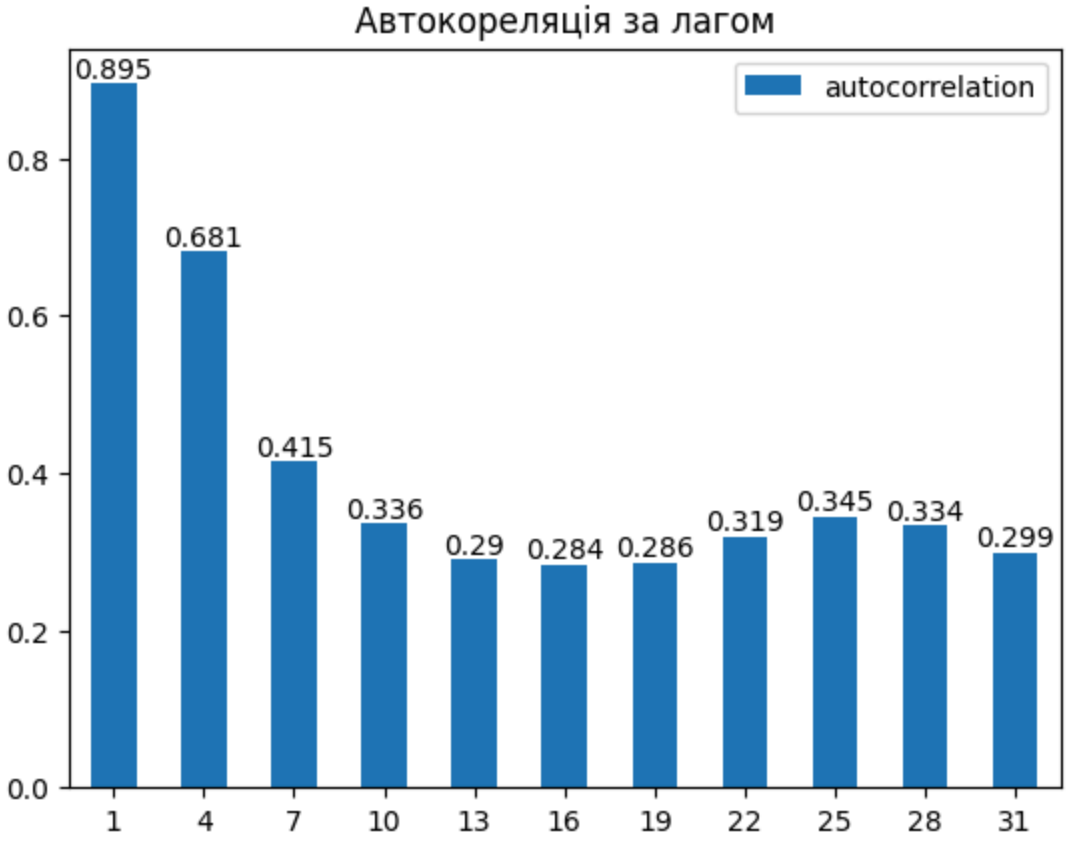
Повортні точки:
{'tanks': 509, '3': 307, '5': 360, '7': 355}

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

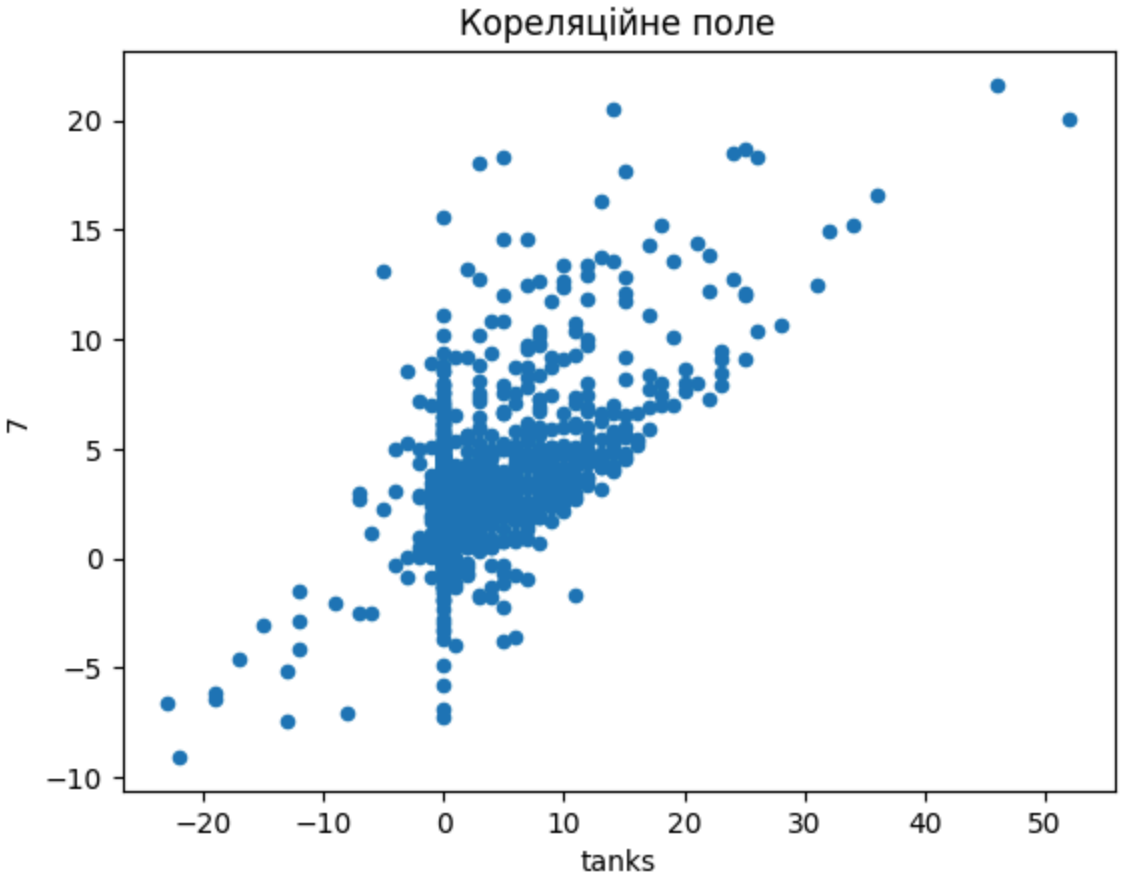

Кореляційне поле
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
7
−20
−10
0
10
20
30
40
50
tanks

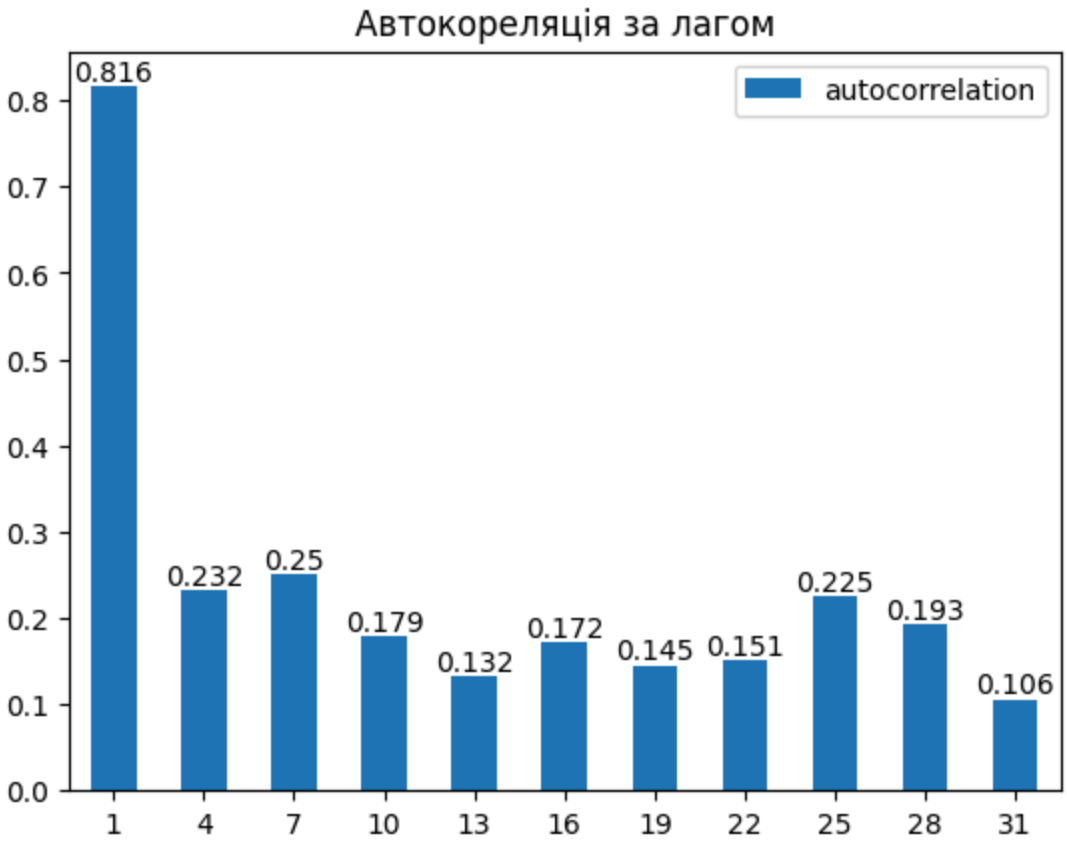

Автокореляція за лагом
autocorrelation
0.895
0.681
0.415
0.336
0.29
0.284
0.286
0.319
0.345
0.334
0.299
0.0
0.2
0.4
0.6
0.8
1
4
7
10
13
16
19
22
25
28
31

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}

Коеффіціент 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
```

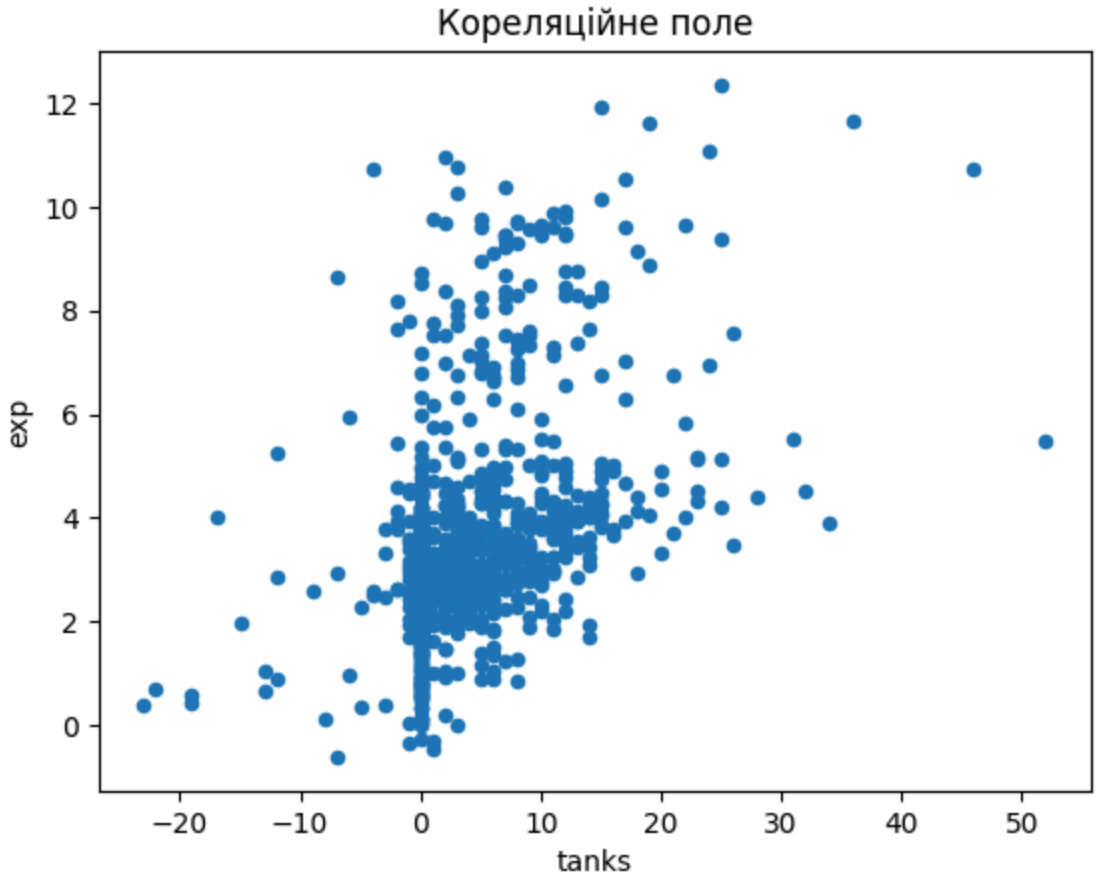


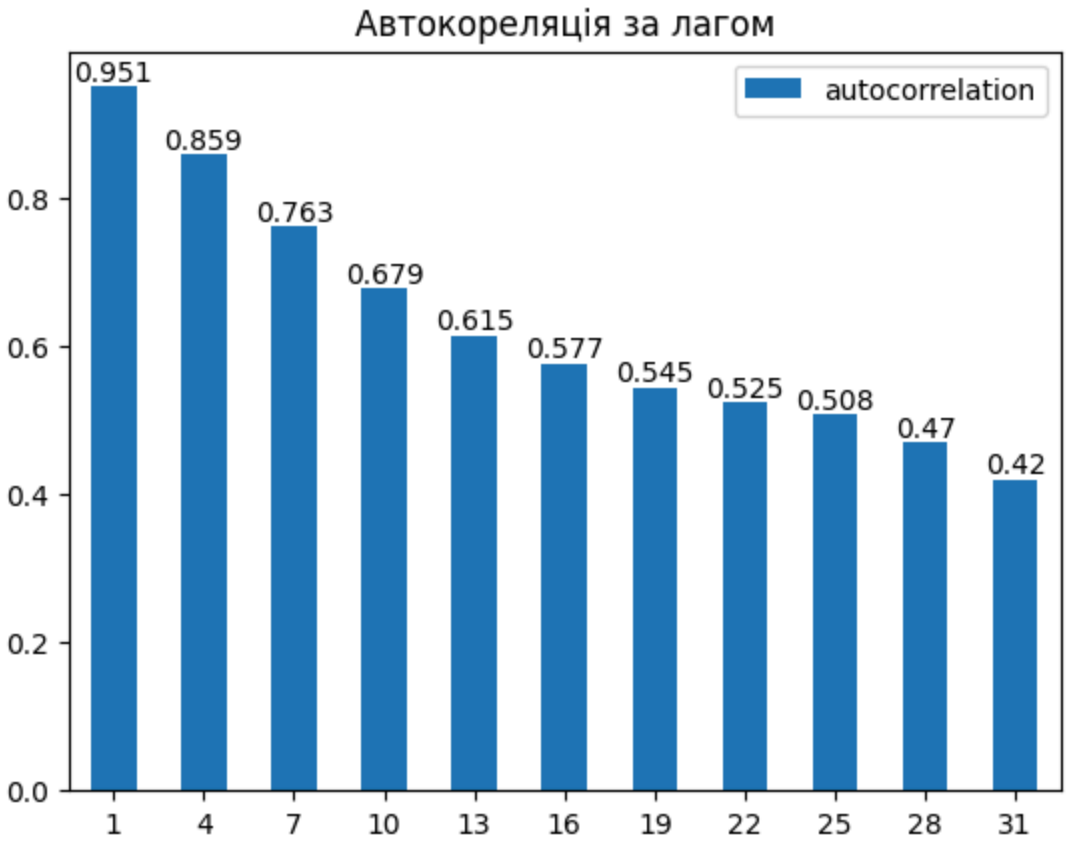


8. Експоненціальне згладжування

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}

Коефіцієнт кореляції між орігінальними даними та згладеними
0.4588239906461523

Кореляція між частинами виборки зі згладеними даними (коефіцієнти множинної
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
```

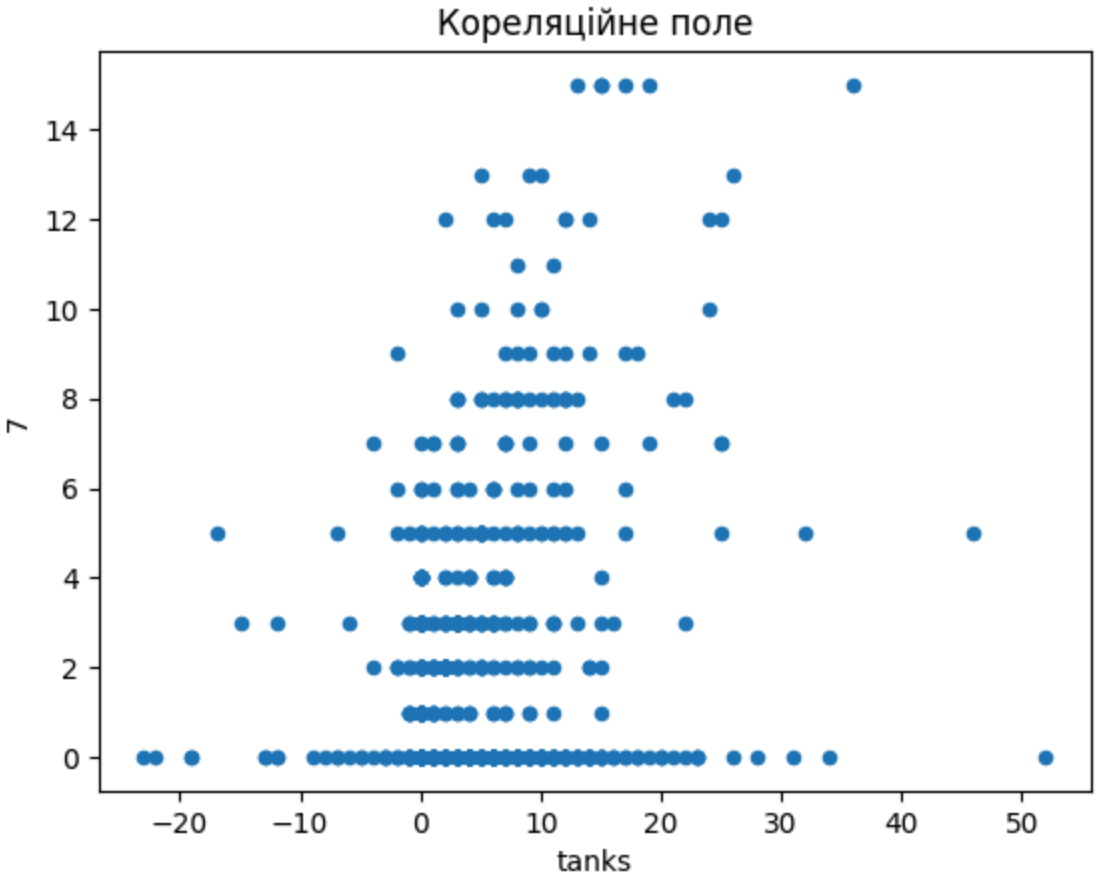


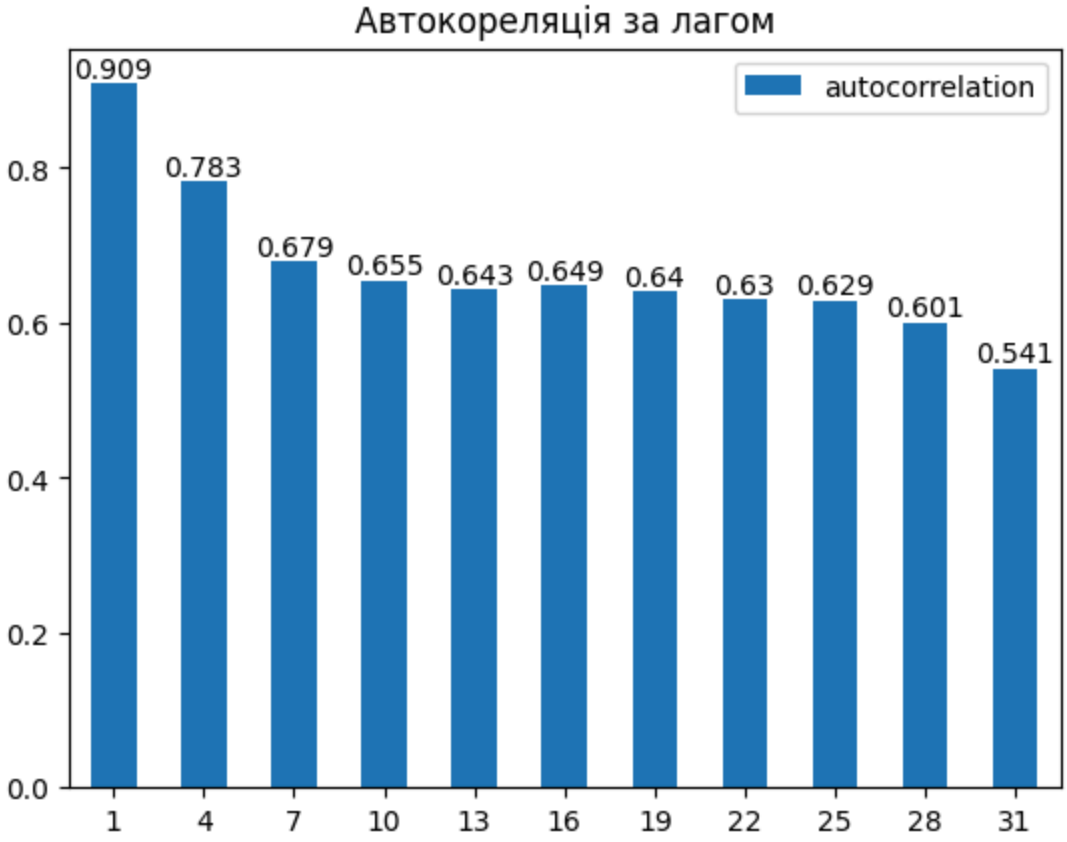


9. Медіанне згладжування

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}

Коеффіціент 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
```


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


Автокореляція за лагом
autocorrelation
0.909
0.783
0.679
0.655
0.643
0.649
0.64
0.63
0.629
0.601
0.541
0.0
0.2
0.4
0.6
0.8
1
4
7
10
13
16
19
22
25
28
31

10. Ієрархічний агломеративний кластерний аналіз багатомірних даних

In [35]:
```
from scipy.cluster import hierarchy as hc
from scipy.spatial.distance import pdist

# зважена відстань на основі кореляції. Використовуємо зважену тому що дані
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()
```