In [ ]:
```
import pandas as pd
df=pd.read_csv("BTC_1sec.csv")
```

In [ ]:
```
df
```

Out[ ]:

	Unnamed: 0	system_time	midpoint	spread	buys	
0	0	2021-04-07 11:32:42.122161+00:00	56035.995	0.01	0.000000	
1	1	2021-04-07 11:32:43.122161+00:00	56035.995	0.01	0.000000	
2	2	2021-04-07 11:32:44.122161+00:00	56035.995	0.01	0.000000	
3	3	2021-04-07 11:32:45.122161+00:00	56035.995	0.01	0.000000	
4	4	2021-04-07 11:32:46.122161+00:00	56035.995	0.01	0.000000	
...	...	...	...	...	...	
1030723	1030723	2021-04-19 09:54:18.386544+00:00	56863.725	0.01	0.000000	
1030724	1030724	2021-04-19 09:54:19.386544+00:00	56863.725	0.01	0.000000	
1030725	1030725	2021-04-19 09:54:20.386544+00:00	56863.725	0.01	1506.866100	
1030726	1030726	2021-04-19 09:54:21.386544+00:00	56863.725	0.01	0.000000	
1030727	1030727	2021-04-19 09:54:22.386544+00:00	56862.275	0.01	66.970689	530

1030728 rows × 156 columns

In [ ]:
```
df['system_time'] = pd.to_datetime(df['system_time'])
```

In [ ]:
```
import pandas as pd
import numpy as np

# Load your LOB data (assuming already loaded as `df`)
# Ensure proper datetime
df['system_time'] = pd.to_datetime(df['system_time'])

# Parameters
window_seconds = 60
step_seconds = 10
min_net_volume = 100  # BTC
min_directionality = 0.9
```

```
metaorders = []

# Ensure data is sorted
df = df.sort_values('system_time').reset_index(drop=True)

# Convert to numpy for speed
timestamps = df['system_time'].values
midpoints = df['midpoint'].values
buys = df['buys'].values
sells = df['sells'].values

i = 0
while i < len(df):
    t_start = df['system_time'].iloc[i]
    t_end = t_start + pd.Timedelta(seconds=window_seconds)

    window_df = df[(df['system_time'] >= t_start) & (df['system_time'] < t_end

    if len(window_df) < 10:
        i += step_seconds
        continue  # Skip sparse windows

    total_buys = window_df['buys'].sum()
    total_sells = window_df['sells'].sum()
    net_volume = total_buys - total_sells
    total_volume = total_buys + total_sells

    if total_volume == 0:
        i += step_seconds
        continue

    directionality = abs(net_volume) / total_volume
    if directionality < min_directionality or abs(net_volume) < min_net_volume
        i += step_seconds
        continue

    midpoint_start = window_df['midpoint'].iloc[0]
    midpoint_end = window_df['midpoint'].iloc[-1]
    delta_p = midpoint_end - midpoint_start

    metaorders.append({
        'start_time': t_start,
        'end_time': t_end,
        'direction': np.sign(net_volume),
        'Q': abs(net_volume),
        'ΔP': delta_p,
        'mid_start': midpoint_start,
        'mid_end': midpoint_end,
        'Q_over_V': abs(net_volume) / total_volume
    })

    i += step_seconds  # Slide the window
```

```
metaorders_df = pd.DataFrame(metaorders)
```

In [ ]:
```
metaorders_df
```

Out[ ]:

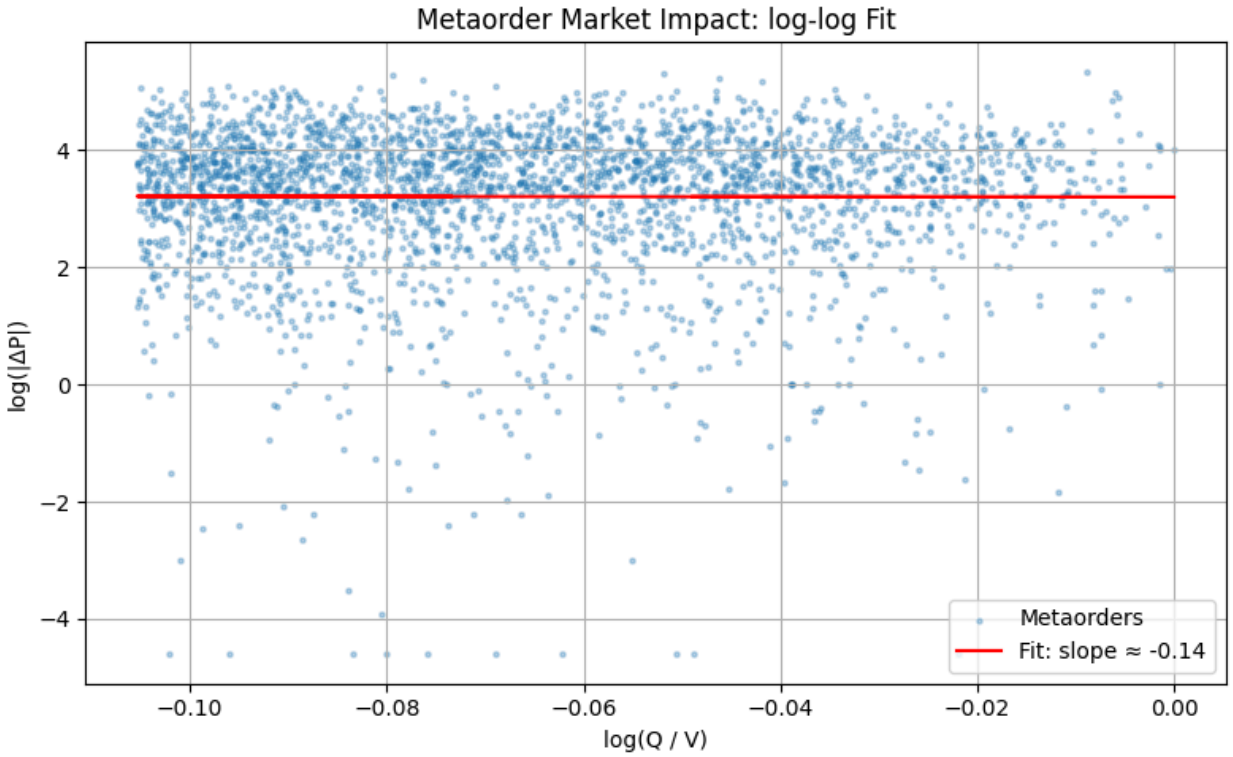
	start_time	end_time	direction	Q	
0	2021-04-07 12:43:12.122161+00:00	2021-04-07 12:44:12.122161+00:00	1.0	9.969273e+05	72
1	2021-04-07 12:55:12.122161+00:00	2021-04-07 12:56:12.122161+00:00	1.0	9.311370e+05	39
2	2021-04-07 13:31:52.122161+00:00	2021-04-07 13:32:52.122161+00:00	1.0	2.003437e+06	57
3	2021-04-07 14:28:32.122161+00:00	2021-04-07 14:29:32.122161+00:00	1.0	6.639931e+05	50
4	2021-04-07 14:28:42.122161+00:00	2021-04-07 14:29:42.122161+00:00	1.0	5.982688e+05	43
...	...	...	...	...	
3039	2021-04-19 09:38:55.386544+00:00	2021-04-19 09:39:55.386544+00:00	-1.0	9.721629e+05	-47
3040	2021-04-19 09:39:05.386544+00:00	2021-04-19 09:40:05.386544+00:00	-1.0	9.433384e+05	-32
3041	2021-04-19 09:39:15.386544+00:00	2021-04-19 09:40:15.386544+00:00	-1.0	1.118498e+06	-56
3042	2021-04-19 09:39:25.386544+00:00	2021-04-19 09:40:25.386544+00:00	-1.0	1.130750e+06	-33
3043	2021-04-19 09:39:35.386544+00:00	2021-04-19 09:40:35.386544+00:00	-1.0	1.125538e+06	-31

3044 rows × 8 columns

In [ ]:
```
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression


df = metaorders_df.copy()
df = df[(df['ΔP'] != 0) & (df['Q_over_V'] > 0)]

# Log-transform
df['log_QV'] = np.log(df['Q_over_V'])
df['log_dP'] = np.log(np.abs(df['ΔP']))

# Fit linear regression
X = df[['log_QV']]
y = df['log_dP']
reg = LinearRegression().fit(X, y)
```

```
slope = reg.coef_[0]
intercept = reg.intercept_

print(f"Fitted power law: |ΔP| = exp({intercept:.4f}) * (Q/V)^{slope:.4f}")

# Plot
plt.figure(figsize=(8, 5))
plt.scatter(df['log_QV'], df['log_dP'], s=5, alpha=0.3, label='Metaorders')
plt.plot(df['log_QV'], reg.predict(X), color='red', label='Fit: slope ≈ {:.2f}
plt.xlabel('log(Q / V)')
plt.ylabel('log(|ΔP|)')
plt.title('Metaorder Market Impact: log-log Fit')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
Fitted power law: |ΔP| = exp(3.1970) * (Q/V)^-0.1372
```



In [ ]:
```
df
```

Out[ ]:

	start_time	end_time	direction	Q	
0	2021-04-07 12:43:12.122161+00:00	2021-04-07 12:44:12.122161+00:00	1.0	9.969273e+05	72
1	2021-04-07 12:55:12.122161+00:00	2021-04-07 12:56:12.122161+00:00	1.0	9.311370e+05	39
2	2021-04-07 13:31:52.122161+00:00	2021-04-07 13:32:52.122161+00:00	1.0	2.003437e+06	57
3	2021-04-07 14:28:32.122161+00:00	2021-04-07 14:29:32.122161+00:00	1.0	6.639931e+05	50
4	2021-04-07 14:28:42.122161+00:00	2021-04-07 14:29:42.122161+00:00	1.0	5.982688e+05	43
...	...	...	...	...	
3039	2021-04-19 09:38:55.386544+00:00	2021-04-19 09:39:55.386544+00:00	-1.0	9.721629e+05	-47
3040	2021-04-19 09:39:05.386544+00:00	2021-04-19 09:40:05.386544+00:00	-1.0	9.433384e+05	-32
3041	2021-04-19 09:39:15.386544+00:00	2021-04-19 09:40:15.386544+00:00	-1.0	1.118498e+06	-56
3042	2021-04-19 09:39:25.386544+00:00	2021-04-19 09:40:25.386544+00:00	-1.0	1.130750e+06	-33
3043	2021-04-19 09:39:35.386544+00:00	2021-04-19 09:40:35.386544+00:00	-1.0	1.125538e+06	-31

2984 rows × 10 columns

In [ ]:

```
df = metaorders_df.copy()
min_dP = 93 # Or adjust depending on your asset tick size / scale
df = df[np.abs(df['ΔP']) > min_dP]
df
```

Out[ ]:

	start_time	end_time	direction	Q	
7	2021-04-07 14:31:02.122161+00:00	2021-04-07 14:32:02.122161+00:00	1.0	1.891924e+06	9
11	2021-04-07 16:53:43.626760+00:00	2021-04-07 16:54:43.626760+00:00	-1.0	7.829543e+06	-12
20	2021-04-07 18:39:43.626760+00:00	2021-04-07 18:40:43.626760+00:00	1.0	3.479805e+06	10
23	2021-04-07 18:48:53.626760+00:00	2021-04-07 18:49:53.626760+00:00	1.0	1.919203e+06	9
25	2021-04-07 18:50:33.626760+00:00	2021-04-07 18:51:33.626760+00:00	1.0	6.806990e+05	9
...	...	...	...	...	
2991	2021-04-19 06:19:14.118284+00:00	2021-04-19 06:20:14.118284+00:00	1.0	3.289756e+05	9
3006	2021-04-19 07:37:24.118284+00:00	2021-04-19 07:38:24.118284+00:00	1.0	1.726055e+06	11
3013	2021-04-19 08:02:15.386544+00:00	2021-04-19 08:03:15.386544+00:00	1.0	1.103097e+05	11
3014	2021-04-19 08:02:25.386544+00:00	2021-04-19 08:03:25.386544+00:00	1.0	1.587680e+05	10
3033	2021-04-19 09:17:35.386544+00:00	2021-04-19 09:18:35.386544+00:00	1.0	1.987653e+05	9

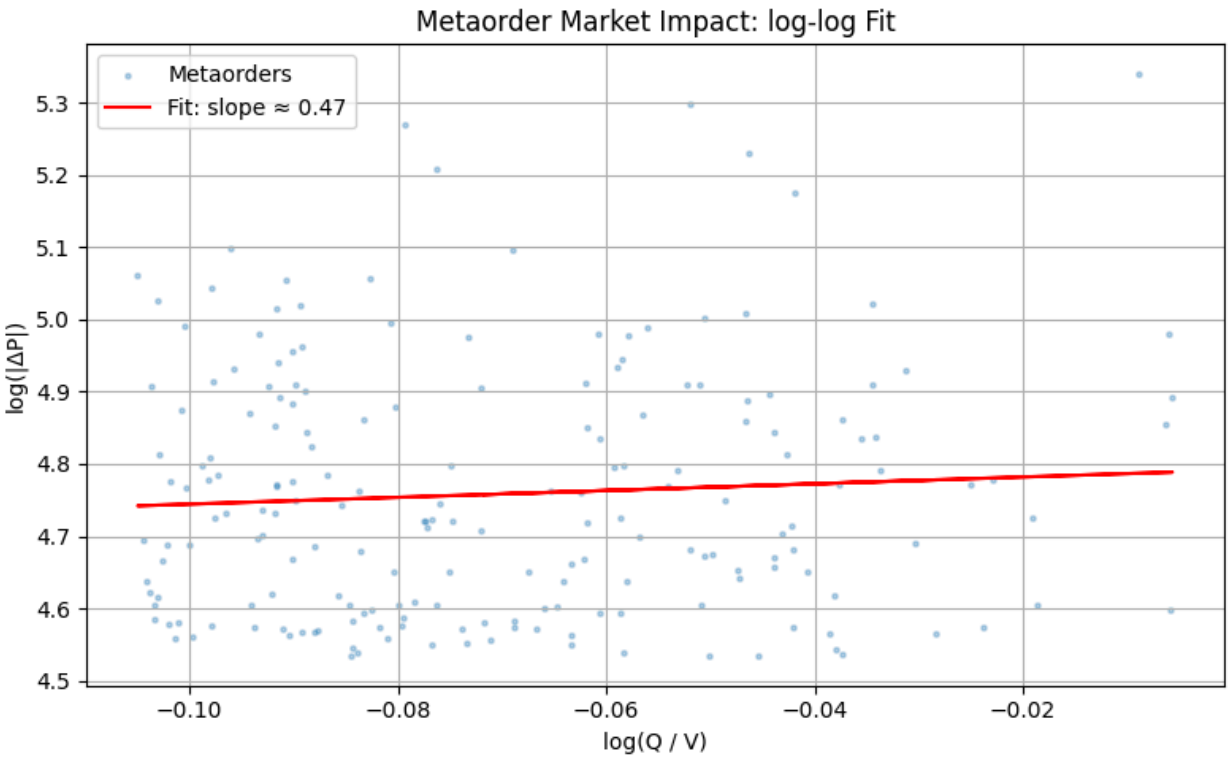
194 rows × 8 columns

In [ ]:

```
# Log-transform
df['log_QV'] = np.log(df['Q_over_V'])
df['log_dP'] = np.log(np.abs(df['ΔP']))

# Fit linear regression
X = df[['log_QV']]
y = df['log_dP']
reg = LinearRegression().fit(X, y)

slope = reg.coef_[0]
intercept = reg.intercept_

print(f"Fitted power law: |ΔP| = exp({intercept:.4f}) * (Q/V)^{slope:.4f}")

# Plot
plt.figure(figsize=(8, 5))
plt.scatter(df['log_QV'], df['log_dP'], s=5, alpha=0.3, label='Metaorders')
plt.plot(df['log_QV'], reg.predict(X), color='red', label='Fit: slope ≈ {:.2f}
plt.xlabel('log(Q / V)')
plt.ylabel('log(|ΔP|)')
```

```
plt.title('Metaorder Market Impact: log-log Fit')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
Fitted power law: |ΔP| = exp(4.7913) * (Q/V)^0.4696
```



In [ ]:
```
df=pd.read_csv("ETH_1sec.csv")
df['system_time'] = pd.to_datetime(df['system_time'])

import pandas as pd
import numpy as np

# Load your LOB data (assuming already loaded as `df`)
# Ensure proper datetime
df['system_time'] = pd.to_datetime(df['system_time'])

# Parameters
window_seconds = 60
step_seconds = 10
min_net_volume = 100  # BTC
min_directionality = 0.9

metaorders = []

# Ensure data is sorted
df = df.sort_values('system_time').reset_index(drop=True)

# Convert to numpy for speed
timestamps = df['system_time'].values
```

```
midpoints = df['midpoint'].values
buys = df['buys'].values
sells = df['sells'].values

i = 0
while i < len(df):
    t_start = df['system_time'].iloc[i]
    t_end = t_start + pd.Timedelta(seconds=window_seconds)

    window_df = df[(df['system_time'] >= t_start) & (df['system_time'] < t_end

    if len(window_df) < 10:
        i += step_seconds
        continue  # Skip sparse windows

    total_buys = window_df['buys'].sum()
    total_sells = window_df['sells'].sum()
    net_volume = total_buys - total_sells
    total_volume = total_buys + total_sells

    if total_volume == 0:
        i += step_seconds
        continue

    directionality = abs(net_volume) / total_volume
    if directionality < min_directionality or abs(net_volume) < min_net_volume
        i += step_seconds
        continue

    midpoint_start = window_df['midpoint'].iloc[0]
    midpoint_end = window_df['midpoint'].iloc[-1]
    delta_p = midpoint_end - midpoint_start

    metaorders.append({
        'start_time': t_start,
        'end_time': t_end,
        'direction': np.sign(net_volume),
        'Q': abs(net_volume),
        'ΔP': delta_p,
        'mid_start': midpoint_start,
        'mid_end': midpoint_end,
        'Q_over_V': abs(net_volume) / total_volume
    })

    i += step_seconds  # Slide the window

metaorders_df = pd.DataFrame(metaorders)
```

In [ ]:
```
metaorders_df
```

Out[ ]:

	start_time	end_time	direction	Q	
0	2021-04-07 11:44:40.861733+00:00	2021-04-07 11:45:40.861733+00:00	-1.0	3.588272e+06	-11
1	2021-04-07 12:42:40.861733+00:00	2021-04-07 12:43:40.861733+00:00	1.0	8.728183e+05	0
2	2021-04-07 13:34:30.861733+00:00	2021-04-07 13:35:30.861733+00:00	1.0	9.284194e+05	2
3	2021-04-07 13:34:40.861733+00:00	2021-04-07 13:35:40.861733+00:00	1.0	7.842454e+05	3
4	2021-04-07 13:34:50.861733+00:00	2021-04-07 13:35:50.861733+00:00	1.0	6.917145e+05	3
...	...	...	...	...	
1434	2021-04-11 02:29:42.009457+00:00	2021-04-11 02:30:42.009457+00:00	1.0	6.182011e+04	-0
1435	2021-04-11 02:29:52.009457+00:00	2021-04-11 02:30:52.009457+00:00	1.0	4.986233e+04	-0
1436	2021-04-11 02:30:02.009457+00:00	2021-04-11 02:31:02.009457+00:00	1.0	4.321305e+04	-0
1437	2021-04-11 02:30:12.009457+00:00	2021-04-11 02:31:12.009457+00:00	1.0	4.087614e+04	0
1438	2021-04-11 02:30:22.009457+00:00	2021-04-11 02:31:22.009457+00:00	1.0	3.825007e+04	0

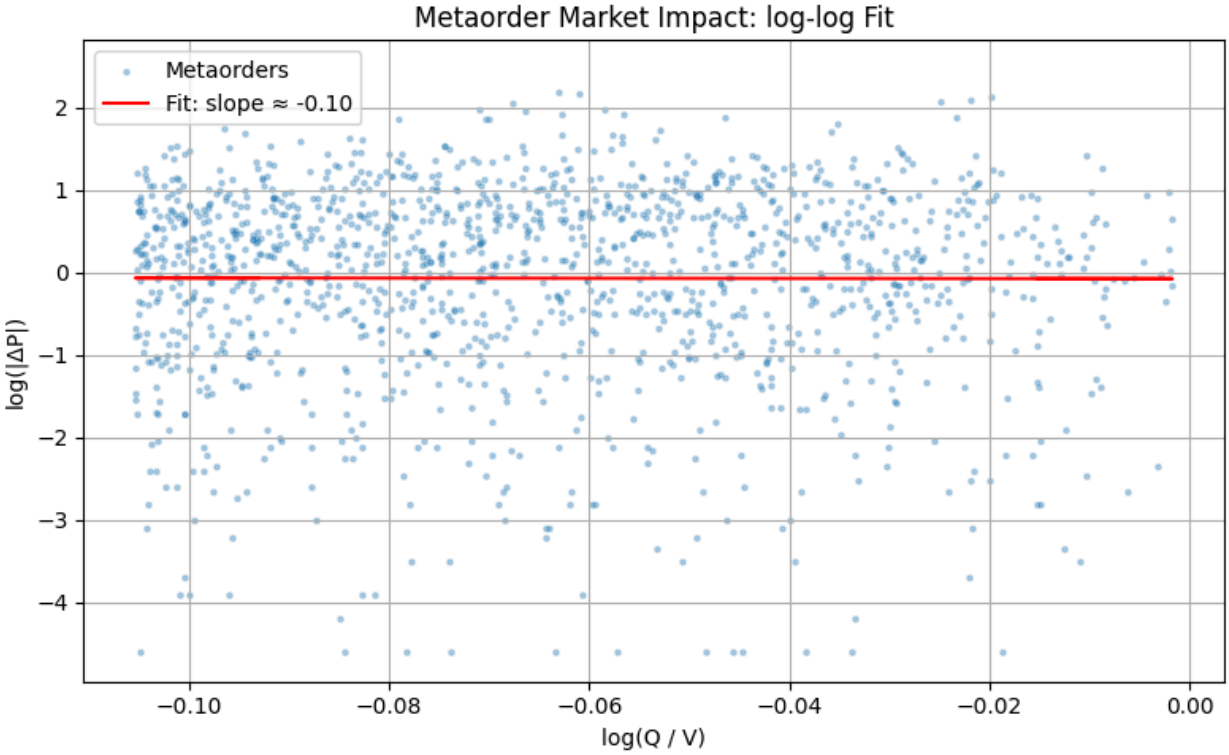
1439 rows × 8 columns

In [ ]:

```
df = metaorders_df.copy()
df = df[(df['ΔP'] != 0) & (df['Q_over_V'] > 0)]

# Log-transform
df['log_QV'] = np.log(df['Q_over_V'])
df['log_dP'] = np.log(np.abs(df['ΔP']))

# Fit linear regression
X = df[['log_QV']]
y = df['log_dP']
reg = LinearRegression().fit(X, y)

slope = reg.coef_[0]
intercept = reg.intercept_

print(f"Fitted power law: |ΔP| = exp({intercept:.4f}) * (Q/V)^{slope:.4f}")

# Plot
plt.figure(figsize=(8, 5))
plt.scatter(df['log_QV'], df['log_dP'], s=5, alpha=0.3, label='Metaorders')
```

```
plt.plot(df['log_QV'], reg.predict(X), color='red', label='Fit: slope ≈ {:.2f}
plt.xlabel('log(Q / V)')
plt.ylabel('log(|ΔP|)')
plt.title('Metaorder Market Impact: log-log Fit')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
Fitted power law: |ΔP| = exp(-0.0739) * (Q/V)^-0.1021
```

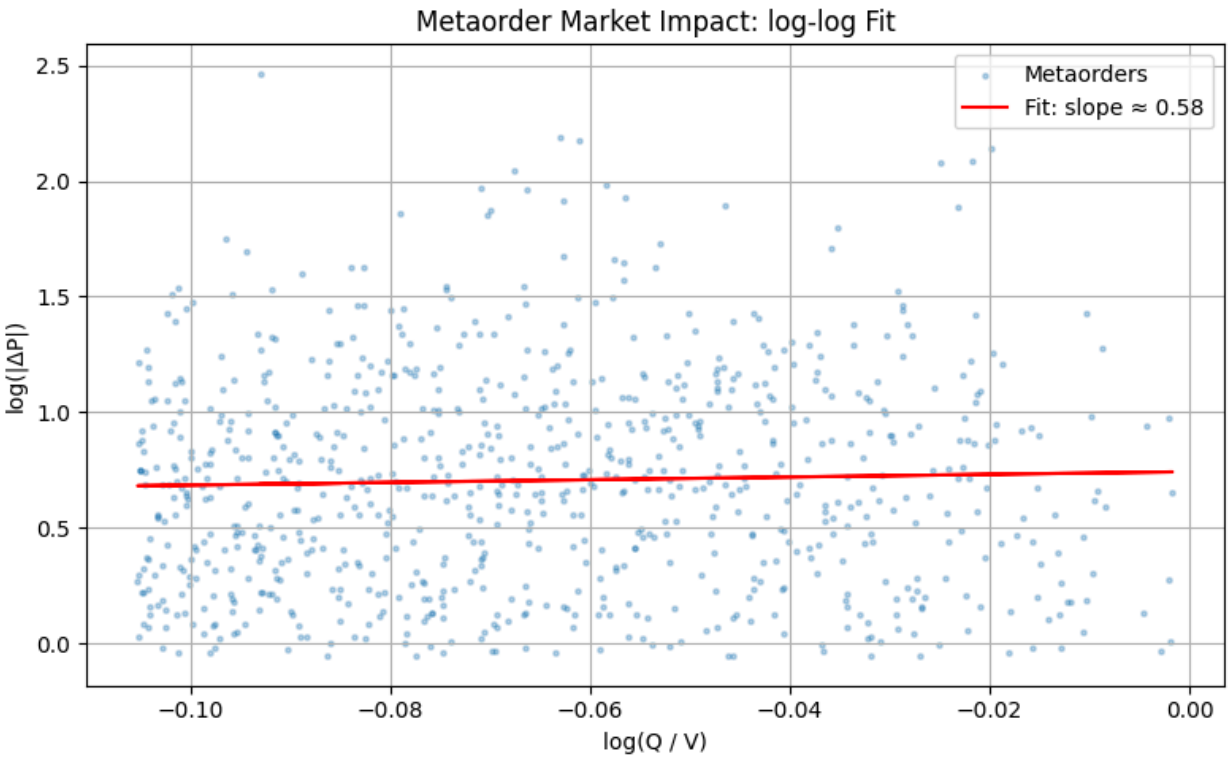


In [ ]:

```
df = metaorders_df.copy()
min_dP = 0.94
df = df[np.abs(df['ΔP']) > min_dP]
df['log_QV'] = np.log(df['Q_over_V'])
df['log_dP'] = np.log(np.abs(df['ΔP']))

# Fit linear regression
X = df[['log_QV']]
y = df['log_dP']
reg = LinearRegression().fit(X, y)

slope = reg.coef_[0]
intercept = reg.intercept_

print(f"Fitted power law: |ΔP| = exp({intercept:.4f}) * (Q/V)^{slope:.4f}")

# Plot
plt.figure(figsize=(8, 5))
plt.scatter(df['log_QV'], df['log_dP'], s=5, alpha=0.3, label='Metaorders')
plt.plot(df['log_QV'], reg.predict(X), color='red', label='Fit: slope ≈ {:.2f}
```

```
plt.xlabel('log(Q / V)')
plt.ylabel('log(|ΔP|)')
plt.title('Metaorder Market Impact: log-log Fit')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
Fitted power law: |ΔP| = exp(0.7427) * (Q/V)^0.5828
```



In [ ]:

```
df=pd.read_csv("ADA_1sec.csv")
df['system_time'] = pd.to_datetime(df['system_time'])

import pandas as pd
import numpy as np

# Load your LOB data (assuming already loaded as `df`)
# Ensure proper datetime
df['system_time'] = pd.to_datetime(df['system_time'])

# Parameters
window_seconds = 60
step_seconds = 10
min_net_volume = 100  # BTC
min_directionality = 0.9

metaorders = []

# Ensure data is sorted
df = df.sort_values('system_time').reset_index(drop=True)
```

```
# Convert to numpy for speed
timestamps = df['system_time'].values
midpoints = df['midpoint'].values
buys = df['buys'].values
sells = df['sells'].values

i = 0
while i < len(df):
    t_start = df['system_time'].iloc[i]
    t_end = t_start + pd.Timedelta(seconds=window_seconds)

    window_df = df[(df['system_time'] >= t_start) & (df['system_time'] < t_end

    if len(window_df) < 10:
        i += step_seconds
        continue  # Skip sparse windows

    total_buys = window_df['buys'].sum()
    total_sells = window_df['sells'].sum()
    net_volume = total_buys - total_sells
    total_volume = total_buys + total_sells

    if total_volume == 0:
        i += step_seconds
        continue

    directionality = abs(net_volume) / total_volume
    if directionality < min_directionality or abs(net_volume) < min_net_volume
        i += step_seconds
        continue

    midpoint_start = window_df['midpoint'].iloc[0]
    midpoint_end = window_df['midpoint'].iloc[-1]
    delta_p = midpoint_end - midpoint_start

    metaorders.append({
        'start_time': t_start,
        'end_time': t_end,
        'direction': np.sign(net_volume),
        'Q': abs(net_volume),
        'ΔP': delta_p,
        'mid_start': midpoint_start,
        'mid_end': midpoint_end,
        'Q_over_V': abs(net_volume) / total_volume
    })

    i += step_seconds  # Slide the window

metaorders_df = pd.DataFrame(metaorders)
```

In [ ]:
```
metaorders_df
```

Out[ ]:

	start_time	end_time	direction	Q	
0	2021-04-07 12:51:00.055697+00:00	2021-04-07 12:52:00.055697+00:00	1.0	30710.153829	0.
1	2021-04-07 14:09:30.055697+00:00	2021-04-07 14:10:30.055697+00:00	1.0	47339.189709	0.
2	2021-04-07 14:09:40.055697+00:00	2021-04-07 14:10:40.055697+00:00	1.0	40996.279816	0.
3	2021-04-07 14:55:20.055697+00:00	2021-04-07 14:56:20.055697+00:00	1.0	27188.391080	-0.
4	2021-04-07 14:55:30.055697+00:00	2021-04-07 14:56:30.055697+00:00	1.0	27045.771153	0.
...	...	...	...	...	
1942	2021-04-11 04:31:52.066133+00:00	2021-04-11 04:32:52.066133+00:00	1.0	34309.508166	0.
1943	2021-04-11 04:32:02.066133+00:00	2021-04-11 04:33:02.066133+00:00	1.0	25543.129164	0.
1944	2021-04-11 04:51:52.066133+00:00	2021-04-11 04:52:52.066133+00:00	-1.0	139887.615558	-0.
1945	2021-04-11 04:52:02.066133+00:00	2021-04-11 04:53:02.066133+00:00	-1.0	140110.284315	-0.
1946	2021-04-11 05:03:37.066133+00:00	2021-04-11 05:04:37.066133+00:00	1.0	47307.519964	0.

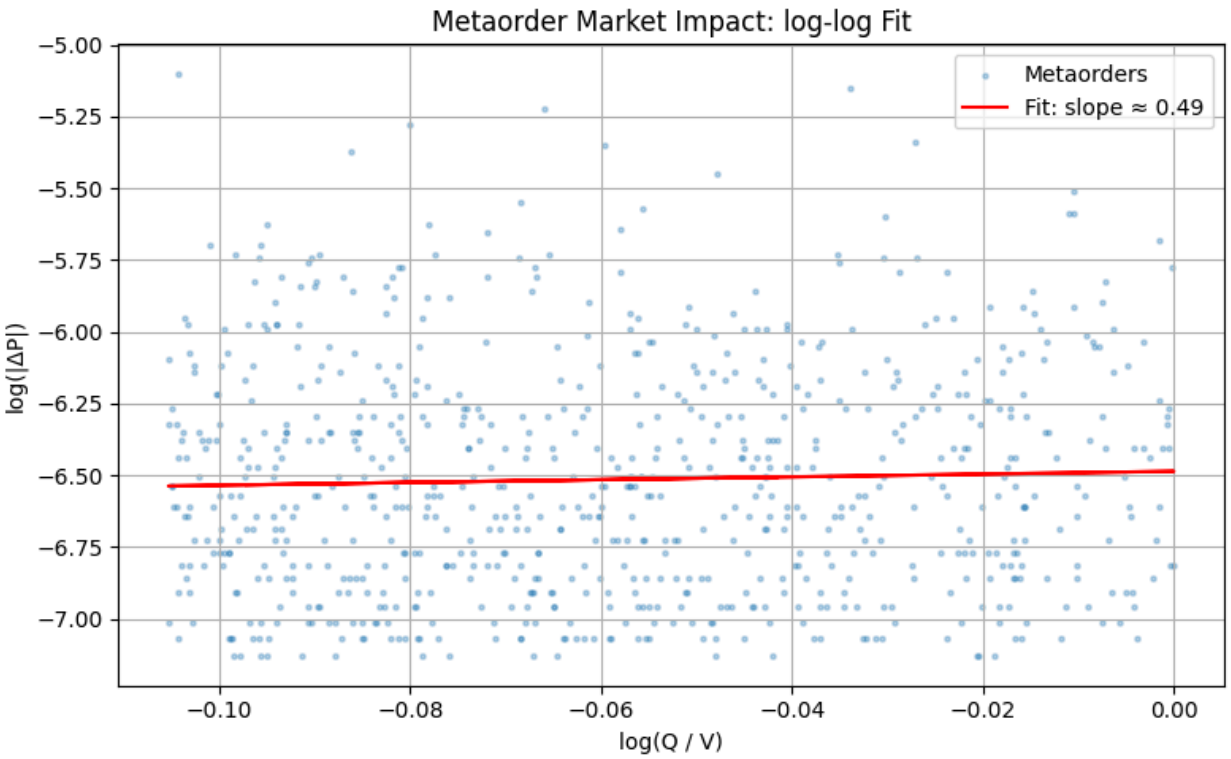
1947 rows × 8 columns

In [ ]:

```
df = metaorders_df.copy()
min_dP = 0.8e-3
df = df[np.abs(df['ΔP']) > min_dP]
df['log_QV'] = np.log(df['Q_over_V'])
df['log_dP'] = np.log(np.abs(df['ΔP']))

# Fit linear regression
X = df[['log_QV']]
y = df['log_dP']
reg = LinearRegression().fit(X, y)

slope = reg.coef_[0]
intercept = reg.intercept_

print(f"Fitted power law: |ΔP| = exp({intercept:.4f}) * (Q/V)^{slope:.4f}")

# Plot
plt.figure(figsize=(8, 5))
plt.scatter(df['log_QV'], df['log_dP'], s=5, alpha=0.3, label='Metaorders')
plt.plot(df['log_QV'], reg.predict(X), color='red', label='Fit: slope ≈ {:.2f}
```

```
plt.xlabel('log(Q / V)')
plt.ylabel('log(|ΔP|)')
plt.title('Metaorder Market Impact: log-log Fit')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
Fitted power law: |ΔP| = exp(-6.4859) * (Q/V)^0.4870
```



In [ ]:
```
import pandas as pd
df_btc=pd.read_csv("BTC_1sec.csv")
df_ada=pd.read_csv("ADA_1sec.csv")
df_eth=pd.read_csv("ETH_1sec.csv")
import numpy as np
```

In [ ]:
```
for i in range (15):
  df_btc[f'bids_distance_{i}'] = df_btc['midpoint']- np.exp(df_btc[f'bids_dist
  df_ada[f'bids_distance_{i}'] = df_ada['midpoint']- np.exp(df_ada[f'bids_dist
  df_eth[f'bids_distance_{i}'] = df_eth['midpoint']- np.exp(df_eth[f'bids_dist
  df_btc[f'asks_distance_{i}'] = df_btc['midpoint']- np.exp(df_btc[f'asks_dist
  df_ada[f'asks_distance_{i}'] = df_ada['midpoint']- np.exp(df_ada[f'asks_dist
  df_eth[f'asks_distance_{i}'] = df_eth['midpoint']- np.exp(df_eth[f'asks_dist
df_ada
```

Out[ ]:

	Unnamed: 0	system_time	midpoint	spread	buys	
0	0	2021-04-07 11:33:00.055697+00:00	1.17075	0.0015	0.000000	0
1	1	2021-04-07 11:33:01.055697+00:00	1.17005	0.0001	684.618694	74178
2	2	2021-04-07 11:33:02.055697+00:00	1.17045	0.0009	1280.056786	1835
3	3	2021-04-07 11:33:03.055697+00:00	1.17005	0.0001	5.487769	1111
4	4	2021-04-07 11:33:04.055697+00:00	1.17005	0.0001	683.701131	712
...	...	...	...	...	...	
323172	322554	2021-04-11 05:09:16.066133+00:00	1.20855	0.0003	8.968554	0
323173	322555	2021-04-11 05:09:17.066133+00:00	1.20870	0.0002	0.000000	0
323174	322556	2021-04-11 05:09:18.066133+00:00	1.20870	0.0002	390.394048	0
323175	322557	2021-04-11 05:09:19.066133+00:00	1.20865	0.0003	0.000000	0
323176	322558	2021-04-11 05:09:20.066133+00:00	1.20865	0.0003	0.000000	0

323177 rows × 156 columns

In [ ]:

```
ts_btc=23000
for j in range (len (df_btc)):
  for i in range (1,15):
    k=i-1
    a=df_btc[f'bids_distance_{i}'][j] - df_btc[f'bids_distance_{k}'][j]
    if (a < ts_btc) and (a>0):
      ts_btc=a

ts_ada=23
for j in range (len (df_ada)):
  for i in range (1,15):
    k=i-1
    a=df_ada[f'bids_distance_{i}'][j] - df_ada[f'bids_distance_{k}'][j]
    if (a < ts_ada) and (a>0):
      ts_ada=a

ts_eth=23
for j in range (len (df_eth)):
  for i in range (1,15):
    k=i-1
```

```
    a=df_eth[f'bids_distance_{i}'][j] - df_eth[f'bids_distance_{k}'][j]
    if (a < ts_eth) and (a>0):
      ts_eth=a

print (ts_eth, ts_ada , ts_btc)
```

```
0.00994409446229838 9.944356781388386e-05 0.009904449863824993
```