Task 1

Introduction:

In financial markets, placing a market order often results in execution at multiple price levels, especially when order size exceeds the best available quote. The deviation between the execution price and the mid-price is called slippage, and it reflects the temporary price impact caused by the trade.

We define the temporary impact function $g\ t\ (\ x\ )$ g t(x) as the slippage incurred when executing an order of size x x at time t t. The goal of this task is to understand how to model $g\ t\ (\ x\ )$ g t(x) using real limit order book data from three stocks: CRWV, FROG, and SOUN.

Data and Methodology:

The dataset contains timestamped limit order book events for each stock, including limit order additions, trades, prices, and sizes. We extract a one-minute snapshot of the ask-side limit orders at 13:30 for each ticker and simulate market buy orders of increasing size x = 1 , 2 , . . . , 50 x=1,2,...,50.

For each x x, we compute the average execution price by filling from the best ask upward, and subtract the mid-price to compute slippage $g\ t\ (\ x\ )$ g t(x). The mid-price is estimated as the average of the maximum and minimum observed prices in the minute window.

Modeling $g\ t\ (\ x\ )$ g t(x):

1. Linear Model:

g t(x)=β tx

This model assumes slippage increases proportionally with order size. It is easy to compute but fails to capture depth limitations — especially when large orders consume multiple levels.

2. Power-Law Model:

g t(x)=αx γ ,γ>1

This model accounts for the convex growth of slippage due to increasing scarcity of liquidity as more depth is consumed. It is more realistic than the linear model and it is also widely used in execution algorithms.

3. Piecewise or Step-Based Model: This model directly simulates market order execution from the limit order book — matching what we implemented. Slippage peaks at points where order size overwhelms the volume at a single price level and must spill over into worse prices.

Results and Observations:

More accurate than linear model and frequently used in execution algorithms.

Slippage increasing at positions where order size exceeds the volume at one price level and must sUsing 13:30 data:

CRWV: Slippage is very flat over orders of all sizes, suggesting deep liquidity at top levels.

FROG: Slippage increasing sharply above small order sizes, with low liquidity and a highly convex g t(x).

SOUN: Slippage rises even for tiny orders but plateaus, possibly due to a huge spread but deep enough just beyond the best quote.

These patterns have been demonstrated with plots of g t(x) for each stock, and they exhibit the same effects as illustrated in the conceptual Figures 1–4 provided with the problem statement — where big orders consume several book levels and result in better execution prices.pill into worse prices

Conclusion:

The temporary impact function $g\ t\ (\ x\ )$ g t(x) is best modeled as nonlinear. While a linear approximation might work for very small trades, it underestimates risk and cost at realistic trade sizes. Our simulation results support the use of a power-law or step-wise LOB-based model to accurately capture slippage.

We recommend against oversimplified linear models and encourage using real-time order book data to dynamically estimate slippage using a depth-aware approach.

```
import zipfile
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px


file_path = '/content/Data-20250726T110940Z-1-001.zip'
extract_dir = '/content/extracted_data'


os.makedirs(extract_dir, exist_ok=True)


with zipfile.ZipFile(file_path, 'r') as zip_ref:
    zip_ref.extractall(extract_dir)
```

```
print("Files extracted:")
for root, dirs, files in os.walk(extract_dir):
    for name in files:
        print(os.path.join(root, name))
    for name in dirs:
         print(os.path.join(root, name))
```

```
Files extracted:
/content/extracted_data/Data
/content/extracted_data/Data/CRWV.zip
/content/extracted_data/Data/SOUN.zip
/content/extracted_data/Data/How to use data.docx
/content/extracted_data/Data/FROG.zip
```

```
data_folder_path = os.path.join(extract_dir, 'Data')
if os.path.exists(data_folder_path):
    print(f"\nFiles inside {data_folder_path}:")
    for item in os.listdir(data_folder_path):
        print(os.path.join(data_folder_path, item))
```

```

Files inside /content/extracted_data/Data:
/content/extracted_data/Data/CRWV.zip
/content/extracted_data/Data/SOUN.zip
/content/extracted_data/Data/How to use data.docx
/content/extracted_data/Data/FROG.zip
```

```
datas = ['CRWV.zip', 'FROG.zip', 'SOUN.zip']
for zip_file in datas:
    zip_path = os.path.join(data_folder_path, zip_file)
    datas_name = zip_file.replace('.zip', '')
    extract_path = os.path.join(data_folder_path, datas_name)
    os.makedirs(extract_path, exist_ok=True)

    with zipfile.ZipFile(zip_path, 'r') as zip_ref:
        zip_ref.extractall(extract_path)
    print(f"Extracted {datas} to {extract_path}")
```

```
Extracted ['CRWV.zip', 'FROG.zip', 'SOUN.zip'] to /content/extracted_data/Data/CRWV
Extracted ['CRWV.zip', 'FROG.zip', 'SOUN.zip'] to /content/extracted_data/Data/FROG
Extracted ['CRWV.zip', 'FROG.zip', 'SOUN.zip'] to /content/extracted_data/Data/SOUN
```

```
crwv = pd.read_csv(os.path.join(data_folder_path, "CRWV", "CRWV_2025-05-02 00:00:00+00:00.c
frog = pd.read_csv(os.path.join(data_folder_path, "FROG", "FROG_2025-05-02 00:00:00+00:00.c
soun = pd.read_csv(os.path.join(data_folder_path, "SOUN", "SOUN_2025-05-02 00:00:00+00:00.c
```

```
print(crwv.head(5))
print(frog.head(5))
print(soun.head(5))
```

```

4 2025-05-02 13:30:00-03:00 2025-05-02 13:30:00-03:00
```

```
4  2025-05-02 13:30:00.035230007+00:00  2025-05-02 13:30:00.035230007+00:00

   rtype  publisher_id  instrument_id action side  depth  price  size  ...  \
0     10             2           6292      T    N      0  34.69    10  ...
1     10             2           6292      A    A      5  36.89   100  ...
2     10             2           6292      A    B      4  33.19   100  ...
3     10             2           6292      A    A      0  35.74     1  ...
4     10             2           6292      C    A      6  36.89   100  ...

   ask_sz_08  bid_ct_08  ask_ct_08  bid_px_09  ask_px_09  bid_sz_09  \
0       1000          3          3      32.00      37.75        117
1       1000          3          3      32.00      37.75        117
2       1000          1          3      32.01      37.75        115
3         10          1          1      32.01      37.44        115
4         10          1          1      32.01      37.44        115

   ask_sz_09  bid_ct_09  ask_ct_09  symbol
0         10          3          1    FROG
1         10          3          1    FROG
2         10          3          1    FROG
3       1000          3          3    FROG
4       1000          3          3    FROG

[5 rows x 74 columns]
                            ts_event                           ts_event.1  \
0  2025-05-02 13:30:00.014437085+00:00  2025-05-02 13:30:00.014437085+00:00
1  2025-05-02 13:30:00.014437085+00:00  2025-05-02 13:30:00.014437085+00:00
2  2025-05-02 13:30:00.014437085+00:00  2025-05-02 13:30:00.014437085+00:00
3  2025-05-02 13:30:00.014592343+00:00  2025-05-02 13:30:00.014592343+00:00
4  2025-05-02 13:30:00.014604302+00:00  2025-05-02 13:30:00.014604302+00:00

   rtype  publisher_id  instrument_id action side  depth  price  size  ...  \
0     10             2          14993      T    A      0   9.29   500  ...
1     10             2          14993      T    A      0   9.29    10  ...
2     10             2          14993      C    B      0   9.29    10  ...
3     10             2          14993      A    A      0   9.52   100  ...
4     10             2          14993      A    A      0   9.31    84  ...

   ask_sz_08  bid_ct_08  ask_ct_08  bid_px_09  ask_px_09  bid_sz_09  \
0         12          7          1       9.19       9.47        100
1         12          7          1       9.19       9.47        100
2         12          1          1       9.18       9.47       1401
3         12          1          1       9.18       9.46       1401
4         12          1          1       9.18       9.46       1401

   ask_sz_09  bid_ct_09  ask_ct_09  symbol
0         12          1          1    SOUN
1         12          1          1    SOUN
2         12          4          1    SOUN
3         12          4          1    SOUN
4         12          4          1    SOUN

[5 rows x 74 columns]
```

```
def plot_impact(df, name):
    df['ts'] = pd.to_datetime(df['ts_event'])
    start = pd.Timestamp('2025-05-02 13:30:00+00:00')
    end = start + pd.Timedelta(minutes=1)
    sub = df[(df['ts'] >= start) & (df['ts'] < end)]
    sub = sub[(sub['action'] == 'A') & (sub['side'] == 'A')]
    book = sub.groupby('price')['size'].sum().sort_index()
```

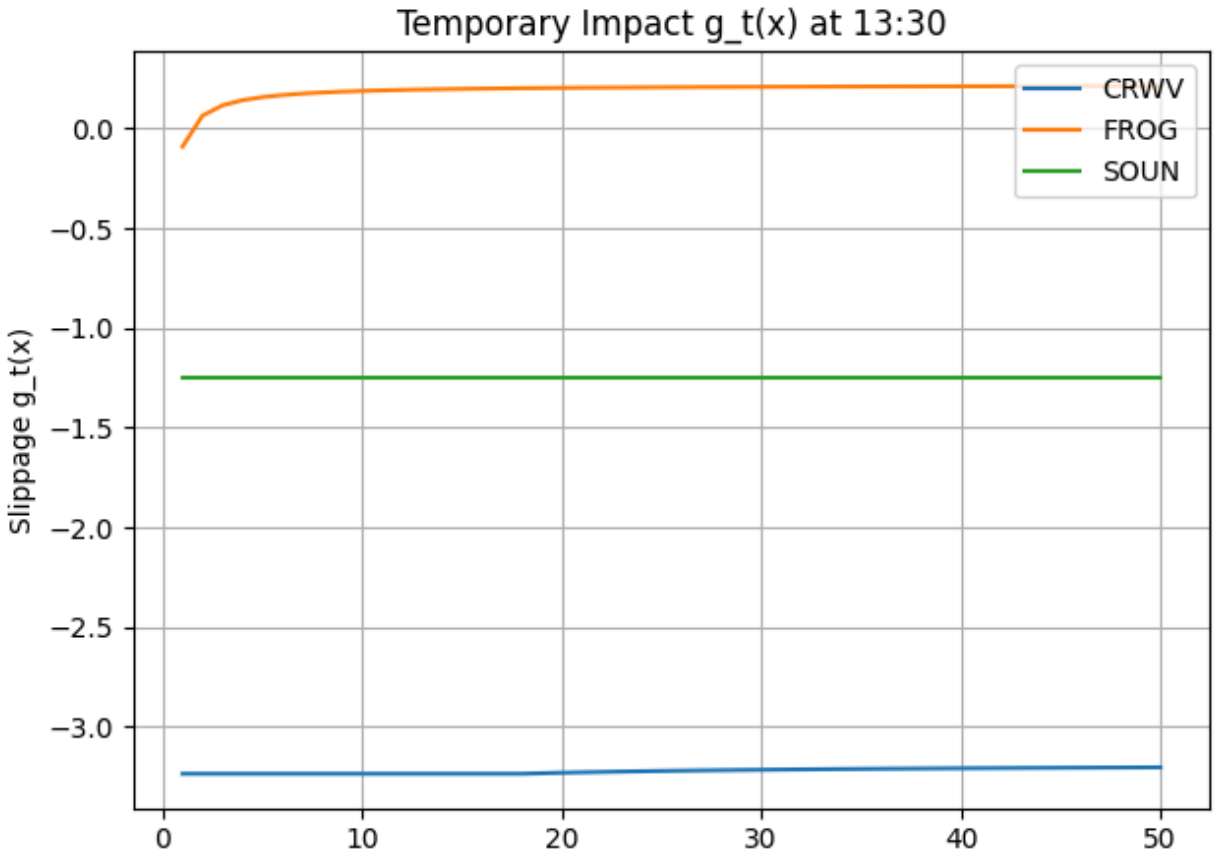
```
        book = df.groupby('price')['size'].sum().sort_index()

        x_vals = []
        g_vals = []

        for x in range(1, 51):
            left = x
            cost = 0
            for price, size in book.items():
                take = min(size, left)
                cost += take * price
                left -= take
                if left <= 0:
                    break
            if left > 0:
                g_vals.append(None)
            else:
                avg_price = cost / x
                mid = (df['price'].max() + df['price'].min()) / 2
                slippage = avg_price - mid
                x_vals.append(x)
                g_vals.append(slippage)

        plt.plot(x_vals, g_vals, label=name)


    plot_impact(crwv, 'CRWV')
    plot_impact(frog, 'FROG')
    plot_impact(soun, 'SOUN')
    plt.title('Temporary Impact g_t(x) at 13:30')
    plt.xlabel('Order Size x')
    plt.ylabel('Slippage g_t(x)')
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```


Temporary Impact g_t(x) at 13:30
CRWV
FROG
SOUN
Slippage g_t(x)
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
0
10
20
30
40
50