Hyperliquid State Price Market Depth Methodology

Hang Miao

Date: October 24, 2025

1. Overview

This document provides guidance for Data Providers (DPs) to compute the Hyperliquid state price consistently and accurately from on-chain limit order book (LOB) data. The goal is to ensure uniformity in methodology and enable faster implementation across our partners.

2. RestAPI

2.1 State Price Endpoints

2.1.1 Shell script to get all state price (mid price):

```
curl -X POST https://api.hyperliquid.xyz/info  \
   -H "Content-Type: application/json" \
   -d '{"type": "allMids"}'
```

2.1.2 The Response Data Format is as follows:


```
{
  "STRAX": "1.5829",
  "STRK": "0.13196",
  "STX": "0.4484",
    ...
  "XRP": "2.46585",
  "YGG": "0.147725",
  "YZY": "0.392475",
  "ZEC": "260.905",
  "ZEN": "12.654",
  "kLUNC": "0.043194",
  "kNEIRO": "0.19916",
  "kPEPE": "0.007231",
  "kSHIB": "0.010245"
}
```

2.2 Limit Order Book Endpoints

2.2.1 Shell script get the current snapshot of limit orderbook:

```
curl -X POST https://api.hyperliquid.xyz/info  \
   -H "Content-Type: application/json" \
   -d '{"type": "l2Book", "coin": "ETH", "nSigFigs": 3}'
```

2.2.2 The Response Data Format is as follows:


```
{
  "coin": "ETH",
  "time": 1760972285920,
  "levels": [
    [
      {
        "px": "4000.0",
        "sz": "21309.1207",
        "n": 1372
      },
      {
        "px": "3900.0",
        "sz": "9728.0976",
        "n": 1253
      },
      ...
      {
        "px": "2200.0",
        "sz": "282.3124",
        "n": 139
      },
      {
        "px": "2100.0",
        "sz": "290.0671",
        "n": 59
      }
    ],
```

```
    [
      {
        "px": "4100.0",
        "sz": "28113.2455",
        "n": 1709
      },
      {
        "px": "4200.0",
        "sz": "13005.866",
        "n": 865
      },
      ...,
      {
        "px": "5900.0",
        "sz": "44.7031",
        "n": 15
      },
      {
        "px": "6000.0",
        "sz": "39.8278",
        "n": 20
      }
    ]
  ]
}
```

3. Websocket

3.1 State Price Subscription

3.1.1 Python script get all state price (mid price):

```
import json
import websocket

def on_message(ws, message):
    data = json.loads(message)
    print(json.dumps(data, indent=2))

def on_open(ws):
    sub_msg = {
        "method": "subscribe",
        "subscription": {
            "type": "allMids",
        }
    }
    ws.send(json.dumps(sub_msg))
    print("Subscribed to all mids")

def on_error(ws, error):
    print("Error:", error)

def on_close(ws, close_status_code, close_msg):
    print("Connection closed")


ws = websocket.WebSocketApp(
    "wss://api.hyperliquid.xyz/ws",
    on_open=on_open,
    on_message=on_message,
    on_error=on_error,
    on_close=on_close
)
ws.run_forever()
```

3.1.2 The Response Data Format is same as from the endpoints


```
{
  "STRAX": "1.5829",
  "STRK": "0.13196",
    ...
  "kPEPE": "0.007231",
  "kSHIB": "0.010245"
}
```


3.2 Limit Orderbook Subscription

3.2.1 Python script get the current snapshot of limit orderbook:

```
import json
import websocket


def on_message(ws, message):
    data = json.loads(message)
    print(json.dumps(data, indent=2))


def on_open(ws):
    sub_msg = {
        "method": "subscribe",
        "subscription": {
            "type": "l2Book",
            "coin": "ETH",
            "nSigFigs": 3,
        },
    }
    ws.send(json.dumps(sub_msg))
    print("Subscribed to l2Book for ETH")


def on_error(ws, error):
    print("Error:", error)
```

```
def on_close(ws, close_status_code, close_msg):
    print("Connection closed")


ws = websocket.WebSocketApp(
    "wss://api.hyperliquid.xyz/ws",
    on_open=on_open,
    on_message=on_message,
    on_error=on_error,
    on_close=on_close,
)
ws.run_forever()
```

3.2.2 The Response Data Format is same as from the endpoints


```
{
  "coin": "ETH",
  "time": 1760972285920,
  "levels": [
    [
      {
        "px": "4000.0",
        "sz": "21309.1207",
        "n": 1372
      },
      ...
      {
        "px": "2100.0",
        "sz": "290.0671",
        "n": 59
      }
    ],


    [
      {
        "px": "4100.0",
```

```
        "sz": "28113.2455",
        "n": 1709
      },
      ...,
      {
        "px": "6000.0",
        "sz": "39.8278",
        "n": 20
      }
    ]
  ]
}
```


4. Market Depth Methodology

In this section, we will introduce the methodology to compute market depth using the raw limit orderbook data ingested from endpoints or websoket.

Let P denotes the current state price (mid price) acquired by the state price endpoints or websocket. It is the mid of top bid ask of the limit orderbook at the highest granularity.

From the response of the limit orderbook endpoints or websockes, the field 'levels' contains two lists: $list_{bids}$ and $list_{asks}$. Each list contains 20 order levels with the following format:

`{'px': '3876.7', 'sz': '58.0147', 'n': 2},`

where

- 'px' is the price level; the px is descending in $list_{bids}$ while ascending in $list_{asks}$. Let $\widetilde{px}_{bids}$ denotes the minimum price level in $list_{bids}$ while let $\widehat{px}_{asks}$ denotes the maximum price level in $list_{asks}$
- 'sz' is the size quoted in base assets. The 'sz' at the i^{th} order level is the accumulated order size over the price range $(\text{px}_{i-1}, px_i]$ when $i \geqslant 1$. For $i = 0$, sz is the accumulated size over $(P, px_0]$ for $list_{asks}$ and $[px_0, P)$ for $list_{bids}$
- 'n' is the number of limit orders.

The parameter 'nSigFigs' $\in \mathcal{Z} : [2, 5]$ in the limit orderbook endpoints and websocket could be used to adjust the orderbook granularity. To ensure that the ingested order book provides sufficient depth to cover at least 1% of the market depth while preserving the highest granularity. The maximum value of 'nSigFigs' such that $\widetilde{px}_{bids} \geqslant 0.99P$ and $\widehat{px}_{asks} \leqslant 1.01P$ should be used.

We could use the state price P to locate the cutoff position for computing the 1% upper and lower market depth as follows:

$$N_{bids} = \arg\max_i px_i \geqslant 0.99P \text{ where } px_i = list_{bids}[i][px] \tag{1}$$

$$N_{asks} = \arg\max_i px_i \leqslant 1.01P \text{ where } px_i = list_{asks}[i][px] \tag{2}$$

With the level cutoffs from 1, 2, we could compute the market depth measure by base token as follows

$$MD_+ = \sum_i^{N_{asks}} sz_i \text{ where } sz_i = list_{asks}[i][sz] \tag{3}$$

$$MD_- = \sum_i^{N_{bids}} sz_i \text{ where } sz_i = list_{bids}[i][sz] \tag{4}$$

The final step is to convert the market depth measured by base value to USD value.

$$USDMD_+ = MD_+ * P \tag{5}$$

$$USDMD_- = MD_- * P \tag{6}$$

where P is the state price. P could also be replaced by other reference base/USD price to enhance robustness.