Rebuilding a limit order book (LOB): Level 1

Tim Meggs - twmeggs@gmail.com

1 Summary

This short paper accompanies some Python functions for reconstituting the event-dependent state of a limit order book from sequential level one[1] tick data.

In and of itself, the algorithm that rebuilds the order book is of little use beyond this specific application; it is not particularly sophisticated but it does the job. The true value of the exercise lies in the realistic representation of the order book's state through time that is achieved. Such a replica can be subjected to analysis. This allows us to extract key statistical features of the price formation process. Examples of the type of analysis that this makes possible are provided towards the end of this paper.

The sections that follow describe the raw input data and outline the steps taken in rebuilding book. They conclude by presenting some examples of order book analysis.

This document should be viewed in conjunction with the Python code that can be found in this Github repo: `https://github.com/twmeggs/PyRebuildLOB`

2 Raw tick data

The raw input information is sequential level one tick data. This includes updates to the best bid and ask, and information on any trades. It looks like this:

[1]In the context of an electronic order book, 'level one' refers to updates of the best bid or ask (price and/or quantity), and to any trades that occur (price and quantity)

Date and Time	side	price	size
06/02/2015 08:00:59	BID	10868	1
06/02/2015 08:00:59	ASK	10869.5	2
06/02/2015 08:00:59	TRADE	10869.5	1
06/02/2015 08:00:59	ASK	10869.5	1
06/02/2015 08:01:00	ASK	10869.5	2
06/02/2015 08:01:00	TRADE	10868	1
06/02/2015 08:01:00	BID	10867.5	1

Each level one event has an associated timestamp (note the dates are given in MM/DD/YYY format). Unfortunately the timestamps are presented with frequency of seconds, while the events they represent occur at times within these seconds (i.e. milli- or nano- second time stamps would be more appropriate). While it would be preferably to have more accurate timestamps, two useful things are known about the data:

1. Events are presented sequentially i.e. each row (order book event) occurred temporally after the preceding row (order book event)

2. For a given timestamp, the event being represented occurred on or after the first instant of that second and before the first instant of the next second e.g. for an event with a timestamp of, say, 08:00:59 that event occurred at some point on or between 08:00:59.000000 and 08:00:59.999999

Given this, we can interpret the data as a sequential-event-stream (regardless of not knowing the exact time at which the events occur) and the events within that stream can be aggregated on a per-second basis should there be a need to do so.

3 From raw data to limit order book replica

This sections outlines the steps taken to go from the raw data to a replica of the order book. Each step is encompassed within a distinct Python function. The working of each function is explained below and they are applied in the order presented, eventually returning a dataframe replica of the order book's state through time.

Python functions:

`df_from_ticks(file_location, index_col)`: A wrapper for `pandas.read_csv` that imports the raw data from `file_location` and builds a time-indexed pandas dataframe. The 'Date and Time' column shown in the table above is set to be the index for the dataframe via `index_col`. Returns a pandas dataframe.

`increase_granularity(df_from_ticks)`: Takes events in `df_from_ticks` that occur within the same second-level timestamp and breaks them out to more granular frequency. Microsecond timestamps are applied. These are arbitrary, but the important sequential character of the events is retained. As long as the microsecond timestamps are not take at the 'true' time at which the events occurred this transformation is harmless but useful. Returns a pandas dataframe.

`build_book(microsecond_ticks)`: (This is a pretty ugly for-loop that could probably be vectorised nicely.) Takes the microsecond granularity dataframe as input. Works through each event and builds a representation of the state of the top-level of the order book AFTER that event has occurred.

Events that cause an update to the state of the order book at a particular timestamp fall into three categories: updates to the best bid price or quantity of contracts being bid at that level (or both); updates to the best ask price or quantity of contracts being offered at that level (or both); trade information, including the quantity traded and at the price of the transaction.

Updating the state of the order book for changes to the information on best bid or ask is relatively trivial. Updating the state after a trade event however involves making a decision around whether to use the trade information to alter the current state of the bid or ask data at that timestamp PRIOR to actually 'receiving' the updated bid or ask message in the event stream. In this implementation of the rebuilding process, it was decided that trade event information would be used to update the bid (or ask) information (size, and potentially price) on the timestamp of the trade. This is consistent with each timestamp representing the state of the order book after the event in question has occurred.

In addition to the four main pieces of level one data (current bid, current bid quantity, current ask, current ask quantity) other information is attached to each timestamp. These are outlined in the next section. Returns a time-indexed dataframe that represents the state of the level one information after each update event has occurred.

The above are the main functions used in building the order book. There are several other Python functions within the code that do a number of things including calculating summary statistics and simulating limit order behaviour.

4 The order book rebuilt

The order book replica is returned from `build_book()` as a pandas dataframe object. Each row (timestamp) has nine columns of information. Current bid, current ask, current bid quantity and current ask quantity are self explanatory (curr_bid, curr_ask, curr_bid_size, curr_ask_size respectively). A boolean is displayed in `trade` to flag timestamps that represent trades; `trade_prc` and `trade_size` give price and size information of the trade that has occurred. The last two columns `last_trade_prc` and `last_trade_side` are carried forward values that supply information on the last transaction to have occurred.

	curr_bid	curr_ask	curr_bid_size	curr_ask_size
2015-06-02 08:00:30.000095	10868.5	10870.5	1	3
2015-06-02 08:00:30.000096	10868.5	10870.5	1	3
2015-06-02 08:00:30.000097	10868.5	10869.5	1	3
2015-06-02 08:00:31.000001	10868.5	10869.5	1	2
2015-06-02 08:00:31.000002	10868.5	10869.5	1	2

	trade	trade_prc	trade_size	last_trade_prc
2015-06-02 08:00:30.000095	True	10868.5	1	10868.5
2015-06-02 08:00:30.000096	False	0	0	10868.5
2015-06-02 08:00:30.000097	False	0	0	10868.5
2015-06-02 08:00:31.000001	True	10869.5	1	10869.5
2015-06-02 08:00:31.000002	False	0	0	10869.5

	last_trade_side
2015-06-02 08:00:30.000095	BID
2015-06-02 08:00:30.000096	BID
2015-06-02 08:00:30.000097	BID
2015-06-02 08:00:31.000001	ASK
2015-06-02 08:00:31.000002	ASK

Having the order book rendered into a dataframe opens up the whole suite of pandas data analysis tools.

5 Some LOB analysis...

Order books present a rich field for study, with many features that can be examined and incorporated into a statistical model. An approach of many recent studies has been to conceptualise the limit order book as a two-sided queueing system, with bid and offer queues interacting with market orders and cancellations[1][2][3]. In such a representation there are many dynamics of interest that warrant study and which need to be correctly specified in any ensuing model.

The below three figures are just three examples of the types of properties that might be of interest, the analysis of which is made possible by an accurate replication of the order book's state, as described above.

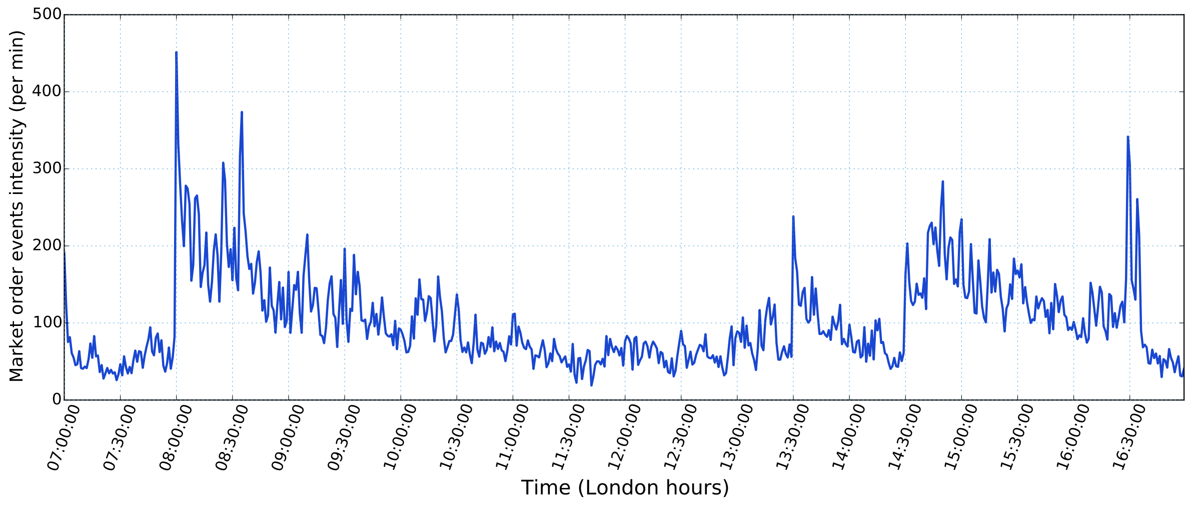

Figure 1: Mean frequency of market orders trades per minute, DAX future

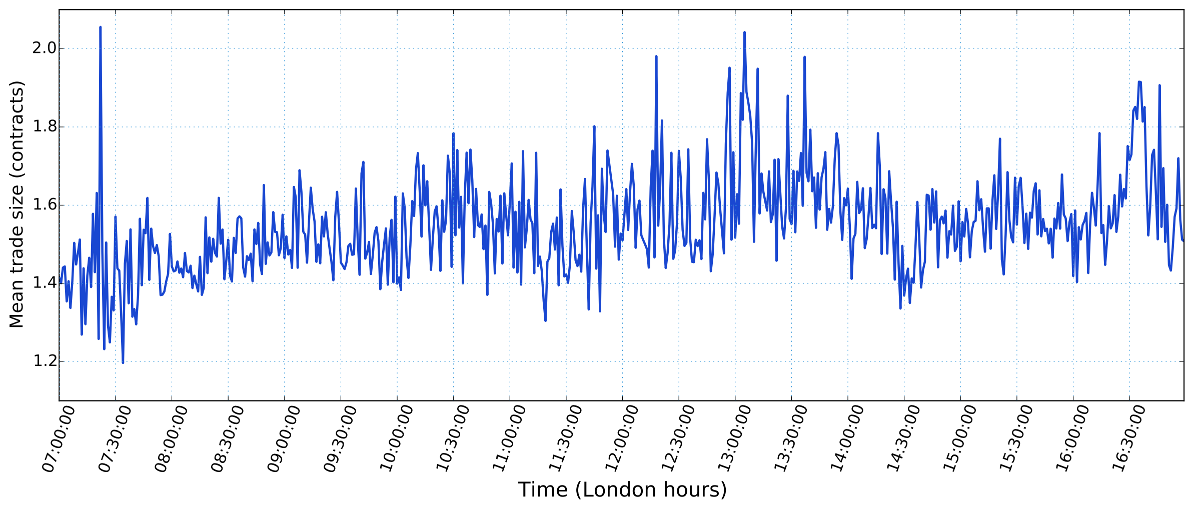

Figure 2: Mean trade size by time, DAX future

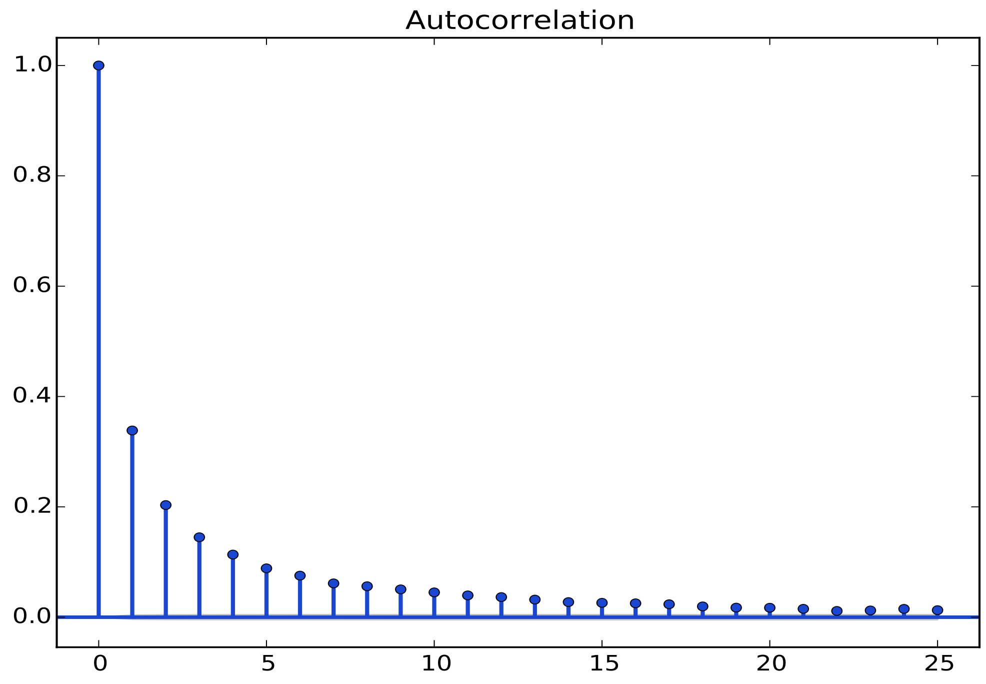

Figure 3: Autocorrelation of trade side (Bid or Ask), DAX Futures

References

[1] E. Smith, J. D. Farmer, L. Gillemot, and S. Krishnamurthy, "Statistical theory of the continuous double auction," *Quant. Finance*, vol. 2, 2003.

[2] R. Cont, S. Stoikov, and R. Talreja, "A stochastic model for order book dynamics," *Oper. Res*, vol. 58, 2010.

[3] R. Cont and A. de Larrard, "Price dynamics in a markovian limit order marker," *SSRN*, 2010. [Online].