The Application of SAS® Hash Object to Ultra-high Frequency Financial Data: A Case Study in Limit Order Book Reconstruction

Dorian M. Noel
The University of the West Indies, St. Augustine, Trinidad and Tobago

ABSTRACT
Over the last decade, the increased availability of ultra-high frequency (UHF) financial data on trades, quotes and order flows in electronic order-driven markets has led to significant innovations in the fields of the empirical microstructure and quantitative analysis of financial markets. Unfortunately, the enormous size of these UHF datasets, usually several million observations per stock per day, has brought up new data processing and handling, in addition to computational challenges.

In today’s trading environment, the key challenge posed by UHF data is how one can efficiently and more importantly, quickly exploit a massive dataset to develop consistent and profitable (alpha generating) algorithmic trading strategies. Central to this exercise is the complete and accurate reconstruction of the electronic limit order book (LOB) of order-driven markets.

This paper attempts to address the research challenge presented by UHF data by demonstrating the application of SAS® Hash Object to the reconstruction of SETS (Stock Exchange Electronic Trading Service) electronic LOB of the London Stock Exchange (LSE).

Keywords: SAS® Hash Object; ultra-high frequency financial data, limit order book, London Stock Exchange.

INTRODUCTION
In recent years, order-driven markets have increasingly made available recorded information (trades, quotes and order flows) from their electronic trading platforms real time (ultra-low latency) to market participants and in the form of UHF datasets to researchers.

The availability of these datasets has led to the phenomenal growth in the applications of statistical and mathematical models to UHF data in order to analyse financial markets. At a theoretical level, these models provide researchers with invaluable insights into the complex interplay between order flow, liquidity and price dynamics. At a more practical level, UHF models support the development of algorithmic order submission strategies and, more recently, the quantitative measurement of key market variables of interest (such as, return, volatility, liquidity and correlation). Further, since the Flash Crash of May 2010, risk practitioners and regulators are more mindful of the fact that proper assessment of risk and market stability ought to be done at a much higher frequency than that considered in the past (typically, daily and monthly). This is mainly due to the recognition of the fact that trading frictions can propagate (as well as, scale) from high frequency to low frequency leading to periods of market instability. Thus, UHF data has also become an invaluable source of market information for regulators and risk practitioners.

However, the sheer size and complexity of UHF datasets constitute a major operational challenge to IT professionals as they are increasingly being called upon to develop new innovative solutions to handle and quickly process these datasets in real time.1 To give some indication of the enormous size of these datasets, the 2008 storage requirements for United States equity trades and quotes are roughly 68,372.7 and 991,443.1 megabytes, respectively.2 For the LSE, these numbers are approximately 4,204.5 and 2,214.5 megabytes, respectively. The size of UHF datasets continue to grow exponentially each year as ultra-low latency algorithmic trading systems have resulted in increased order submissions in less than a millisecond. Indeed, the LSE boasts that SETS executes millions of trades a day at millisecond latencies.

The main objective of the paper is to demonstrate the application of SAS® Hash Object to the reconstruction of SETS electronic LOB of the LSE from its UHF data. The practical use of the approach is that both buy- and sell-side firms can adopt the method to develop UHF algorithmic trading systems. Sewell and Yan (2008), however,

---

1 Keintz (2010) provides a SAS programme solution to efficiently and quickly read UHF datasets.
2 These numbers represent the storage size requirements for compressed ASCII files (see Nexa Technology 2009).
note that UHF algorithmic trading systems need to be combined with ultra-low latency market access if one wishes to utilise these systems to consistently generate alpha in today’s fiercely competitive financial markets.

LIMIT ORDER BOOK: DEFINITION AND CHARACTERISTICS

In many of the world’s most important equity markets such as NYSE Euronext, LSE, Deutsche Börse and Tokyo Stock Exchange, an electronic order-driven system is employed for the trading of the most liquidity securities listed on these exchanges. In such a trading system, trades are a result of the matching of public orders.

Traditionally, traders can use one of the two archetypal order types to indicate their buying or selling interests: (1) market order or (2) limit order. A market order indicates an interest to immediately buy or sell a given number of shares at the best available price in the market. In addition to specifying an order size, a limit order also states a limit price, which indicates the worst price a trader is willing to accept for his order. In the case of a buy limit order (bid), it indicates the maximum price the trader is prepare to pay for his orders and for a sell order (ask), it states the minimum price the trader is willing to accept.

The imbalance between market (liquidity-demanding) and limit (liquidity-supplying) orders is stored in what is commonly referred to as a “limit order book”. In most financial markets, only liquidity–supplying orders are added to the LOB. The LOB, therefore, is simply an inventory list of all buy and sell limit orders (the various bid and ask prices, with their respective sizes) which await execution. On a more technical level, it represents the market’s expressed demand and supply for a stock at a particular point in time.

Orders stored in the LOB are ranked in accordance with price and then, time priority (FIFO rule). For the buy (sell) side of the LOB, the ranking is from the highest (lowest) to the lowest (highest) price. If two limit orders have a similar limit price, then the one with the earlier timestamp is positioned ahead of the other. Figure 1 shows a schematic representation of a LOB.

Figure 1: A Schematic Representation of a Limit Order Book

The state of the LOB changes throughout the trading day in response to traders’ messages. There are four types of messages that traders can submit to a LOB: add, cancel, cancel/replacement, and market order. Add messages add new limit orders to the LOB, while cancel messages delete existing orders from the LOB. If a trader wants to

---

3 Although the types of orders available to traders have grown substantially in recent years, most of these new order types are based on the generic properties of market (demand of immediacy) and limit orders (demand for price protection). See Harris (2004) for an excellent discussion on the most common types of orders found in financial markets.

4 On a typical trading day, the vast majority of the activity in the LOB consists of add and cancel messages as traders aggressively compete for price and/or time priority. The matching of orders is a distant third.
reduce the size of her order, she can issue a cancel/replace message, which cancels the order, then immediately replaces it with another order at the same price, but with a lower size. A trade occurs in the LOB when an incoming aggressively priced limit order or market order “hits” a limit order on the other side of the market. However, only the unfilled portions of limit orders are added to the LOB.

Every order that is submitted to the LOB is recorded, time stamped and given a unique ID. Therefore, the entire event history of an order in LOB can be traced that is, its time of arrival, cancellation or trade matching with other orders. The timestamp of an order determines its time priority in the LOB. Earlier orders are executed before later orders with similar limit price. Figure 2 provides a graphical representation of the dynamic evolution of a LOB during a typical trading day.

Figure 2: A Schematic Representation of the Evolution of the Limit Order Book
The figure below shows a graphical representation of the dynamic evolution of a LOB during a typical trading day. The sell ("Ask") LOB is denoted by the green line, while buy ("bid") by the blue line. The evolution of the LOB is driven largely by the arrival rates of buyer/seller initiated trades, order cancellations and order additions. After each activity (that is, trades, cancellation or addition) in the LOB, outstanding volume (depth) at each price level is recalculated and price and time priority of outstanding orders are re-established. In the figure, each line represents a price level and a price level appears as the result of orders being submitted to the LOB. Similarly, a price level may disappear due to trades or order cancellations.

As previously mentioned, the LOB data, inter alia, is the primary source of market information for algorithmic trading systems and market observers. It is important, therefore, to ensure that computer programmes written to reconstruct the LOB are both unequivocally accurate and extremely fast. The ultimate goal is to re-create the LOB in O(1) time making it possible for a trader to ask questions such as:

1. "What are the best prices in the LOB?"
2. "What is the current quoted price of liquidity in the LOB?"
3. "What is the likely price impact of a market order with size X?"
4. "What is relative position of my order from top of the LOB?"

Since each event that occurs in the LOB is recorded, time stamped and given a unique ID and the rules of order matching (that is, price and time priority) are clear, a hash table lends itself as a natural programming solution to LOB reconstruction. In the next section, we demonstrate the application of SAS® Hash Object to the reconstruction of SETS LOB. We refer readers to Secosky and Bloom (2007) and Dorfman and Snell (2003) for a gentle introduction to SAS® Hash Object. Sedgewick (1988, Chapter 16) provides an expertly written discussion on hashing algorithms.

THE SAS® HASH OBJECT ROUTINE FOR LOB RECONSTRUCTION
In this section, we show how to reconstruct the SETS LOB of the LSE using its UHF dataset, Transaction Data Service (TDS), provided to researchers. We use the Order History (OH) and Snits Orders (SO) files of TDS to
recreate the entire LOB for Compass Group Plc (Ticker: CPG) for period from December 2, 2002 to February 28, 2003, a total of 62 trading days. Jointly, both files contain the necessary order history and transactions information that are needed to rebuild the LOB for any stock trading on SETS.

The OH records all events associated with the life of an order from the time that it is entered to the time that it is removed from the LOB either through execution, deletion or expiration. Every event is time-stamped and since all transactions are recorded electronically, the exact sequence of the order flow is unambiguous. Hence, we know the time each order was entered, its size and type. We also know the time the order was matched against another order on the LOB and the size of the resulting trade. However, information on the limit prices of orders is not recorded in the OH. Such information is contained in the SO. The SO only records information on new orders submitted to SETS.

In Figure 3, we provide some order data extracted from a read of the OH for CPG, while Figure 4 shows extracted data from SO file.

We highlighted three distinct orders in Figure 3 and Figure 4 in order to give a description of the structure of the information contained in OH and SO. The order highlighted in yellow is a new ("N") buy ("B") limit order ("LO") for 5,000 units ("aggsize") of CPG stock submitted to the LOB on 2 December 2002 at 10:26:59 am. The limit price of the order is recorded in the SO (see Figure 4). The order is recorded with the same unique order ID ("ohecode") in both files.

The order highlighted in green is a fill or kill order ("AB") to buy ("B") 10,266 units of stock, which is immediately executed in-full ("M") upon submission to the LOB. The order generated multiple trades, partial ("P") matching with order "502J1DHH02" for 7,614 stock units and then matched with order "502J1CKV02" for the remaining stock units. Finally, the blue highlighted order is an aggressively priced (low selling price) limit order, which is immediately executed in-full upon submission. As a result, the order is not stored in the LOB.

Figure 3: Extract from the Order History (ViewOHe.txt)
The key data variables needed from this file for the reconstruction exercise are: (1) the order code ID ("ohecode"); (2) order type ("ohetype"); (3) event history type ("histype"); (4) order sign ("ohebs") so as to reconstruct the buy/sell LOB; (5) the size of an order removed from LOB after each event ("trasize"); the remaining size of an order after each event ("aggsize") and (5) the time of each order event ("event_date").
The dataset shown in Figure 5 above is the input data for the reconstruction of the LOB. For programming ease, we re-label order event types ("histype") so that we can track the orders that will eventually become part of the LOB. The SAS code that does this is shown below.
Unfortunately, not all the data fields associated with an order are fully populated in the OH. Missing data may cause a programme not to run. The SAS code that populates missing data fields is as follows:

```sas
/* Re-labelling order event types ("HistType") on the Order History */
data book (keep = event_date lo_price ohecode ohetype ohebs aggsize histype order_date trasize ordsize);
  set trades.cpgohist;
  if histype = 'N' then histype = 'A';
  else if histype = 'P' then histype = 'B';
  else if histype in ('M','D','E','F','C','T') then histype = 'D';
run;

/* Filling missing fields in the Order History.*/
proc sort data = book;
  by ohecode event_date descending ohetype;
run;

data book ( drop = _ltype_ _lbs_);
  length _ltype_ _lbs_ $4;
  _ltype_ = ' '; _lbs_ = ' '; 
  do until (last.ohecode);
    set book;
    by ohecode event_date descending ohetype;
    if ohetype = '' then ohetype = _ltype_; 
    _ltype_ = ohetype;
    if ohebs = '' then ohebs = _lbs_; 
    _lbs_ = ohebs;
    output;
  end;
run;

After each event time in the LOB, we need to keep track of the amount of stock units of an order that is removed from the LOB due to either matching with incoming orders or cancellation. Since an order can generate multiple trades at the same timestamp, we simply aggregate the volume for each order and take the last event for an order at each timestamp. The following code completes this task.

```sas
/* Take the last order event at each timestamp, summing the executed amount of each order.*/
proc sort data = book;
  by event_date ohecode histype descending aggsize;
run;

data book;
  do travol = 0 by 0 until (last.ohecode);
    set book;
    by event_date ohecode;
    travol + trasize;
  end;
run;
```
We also need to keep track of the number of new orders and their sizes to be added to the LOB after each event time in the LOB. Similarly, the number of orders and volume to be removed due to either matching or cancellation need to be determined. This exercise is completed by the following programme codes.

/* Keeping track of new orders to be added and their size, executed size of stored orders and the size of order cancelled.*/

```data` book (keep = event_date lo_price ohebs tamt flag);
  set book;
  if event_date = order_date and (histype in ('A','B')) then do;
    flag = 1;
    tamt = aggsize;
  end;
  else if event_date = order_date and histype = 'D' then do;
    flag = 0;
    tamt = aggsize;
  end;
  else if event_date > order_date and histype = 'B' then do;
    flag = 0;
    tamt = -travol;
  end;
  else if event_date > order_date and histype = 'D' then do;
    if ohetype NE 'MO' then do;
      flag = -1;
      tamt = -travol;
    end;
    else do;
      flag = -1;
      tamt = -ordsize;
    end;
  end;
run;

/* Aggregating order additions and deletions at each event timestamp, price level and buy/sell LOB.*/

```proc` sort data = book;
  by event_date lo_price ohebs;
run;
```data` book (drop = flag tamt);
  do until (last.event_date);
    stamt = 0;
    sflag = 0;
    do until (last.ohebs);
      set book;
      by event_date lo_price ohebs;
      stamt += tamt;
      sflag += flag;
    end;
  end;
run;
```

After the above programmes are run, we obtain a dataset which summarises the number of orders and aggregated order volume to be added or deleted from the buy or sell LOB at each price level after each event in

Figure 6: Input Data for the Hash Programme

<table>
<thead>
<tr>
<th>stamt</th>
<th>sflag</th>
<th>ohebs</th>
<th>event_date</th>
<th>lo_price</th>
</tr>
</thead>
<tbody>
<tr>
<td>5000</td>
<td>1</td>
<td>B</td>
<td>02Dec2002:10:26:59</td>
<td>313.25</td>
</tr>
<tr>
<td>-800</td>
<td>-1</td>
<td>S</td>
<td>02Dec2002:10:27:00</td>
<td>313.5</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>B</td>
<td>02Dec2002:10:27:04</td>
<td>313.25</td>
</tr>
<tr>
<td>-5000</td>
<td>-1</td>
<td>B</td>
<td>02Dec2002:10:27:04</td>
<td>313.25</td>
</tr>
<tr>
<td>8679</td>
<td>1</td>
<td>S</td>
<td>02Dec2002:10:27:04</td>
<td>313.25</td>
</tr>
<tr>
<td>-10266</td>
<td>-2</td>
<td>S</td>
<td>02Dec2002:10:27:04</td>
<td>313.5</td>
</tr>
<tr>
<td>3300</td>
<td>1</td>
<td>S</td>
<td>02Dec2002:10:27:04</td>
<td>313.75</td>
</tr>
<tr>
<td>-13679</td>
<td>-1</td>
<td>S</td>
<td>02Dec2002:10:27:04</td>
<td>314.0</td>
</tr>
<tr>
<td>-3300</td>
<td>-1</td>
<td>S</td>
<td>02Dec2002:10:27:06</td>
<td>313.75</td>
</tr>
<tr>
<td>-8679</td>
<td>0</td>
<td>B</td>
<td>02Dec2002:10:27:07</td>
<td>313.0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>S</td>
<td>02Dec2002:10:27:07</td>
<td>313.75</td>
</tr>
<tr>
<td>-8679</td>
<td>-1</td>
<td>S</td>
<td>02Dec2002:10:27:07</td>
<td>313.25</td>
</tr>
<tr>
<td>3300</td>
<td>1</td>
<td>S</td>
<td>02Dec2002:10:27:07</td>
<td>313.75</td>
</tr>
</tbody>
</table>
the LOB (see Figure 6). The resulting data has a structure that allows us to a hash table (SAS® Hash Object) to reconstruct the LOB of CPG. The structure of the programme is provided below.

/* Hashing programme to reconstruct the both sides of the LOB after each order event. For each price line, the reconstructed LOB contains the aggregated volume and the number of orders outstanding.*/
data lob (keep = event_date lo_price ohebs depth ordcnt);
/* initialization */
if _n_ = 1 then
  do;
    set book point = _n_
    length depth 8 ordcnt 8;
    declare hash hh (hashexp: 10);
    /* Set up the hash table to recreate outstanding volume and number of orders at each price line in the buy and sell LOB*/
    hh.DefineKey ('lo_price', 'ohebs');
    hh.DefineData ('lo_price', 'ohebs', 'depth', 'ordcnt');
    DefineDone ();
    call missing (lo_price, ohebs, depth, ordcnt);
    declare hiter hiter ('hh');
  end;
  do until (last.event_date);
    set book;
    /* The objective is to reconstruct the LOB after each event time.*/
    by event_date;
    rc = hh.find ();
    /* If buy or sell limit price is found in the table, add aggregated size of the new orders to outstanding volume and add the number of new orders to number of outstanding orders at each price line.*/
    if (rc = 0) then
      do;
        depth ++ stamt;
        ordcnt ++ sflag;
        hh.replace ();
      end;
    else
      do;
        depth = stamt;
        ordcnt = sflag;
        hh.add ();
      end;
    end;
    /* After each event time dump the content of the hash table.*/
    rc = hiter.first();
    do while (rc = 0);
      /* If the a table address is empty or size of the remaining order is zero then don't dump the table address.*/
      if depth GT 0 and ordcnt GT 0 then output;
      rc = hiter.next ();
    end;
  run;

The programme is quite fast and efficient in reconstructing the LOB for the sample stock. For the 62 trading days of data we study, the programme takes roughly 18.04 real time seconds (12.48 seconds CPU time) to reconstruct

---

5 This excludes the reading and processing of the data files, which take roughly 8 minutes to complete for CPG. We run our programme on a standard desktop computer with a Pentium 4 processor, 3.4 GHz of clock speed and 2 gigabytes of RAM. Nonetheless, the speed of programme depends on the number of uniquely submitted limit prices (hence, the price volatility), as well as the number of order events.

---
all the LOBs that existed during the order management, auction and continuous trading sessions (a total of 190,828 distinct LOBs). Further, we are able to obtain all price levels on bid and ask sides of the LOB and for each price level, the available depth (aggregated outstanding volume) and the size of the waiting queue (the number of outstanding orders). In Figure 7, we provide extracts of the reconstructed LOB of CPG on December 2, 2002 at 10:27:06 GMT and 10:27:07 GMT, respectively.

**Figure 7: Extracts from the Reconstructed LOB of CPG**

The figure shows extracts from the reconstructed LOB of CPG on December 2, 2002. The extract on the left is the reconstructed LOB at 10:27:06 GMT, while the extract on the right shows the LOB at 10:27:07 GMT.

From Figure 6, at event time “02Dec2002:10:27:27”, 8,679 stock units are to be removed from the buy LOB at price level 313 pence. One cancelled sell order of 8,679 stock units at price level 313.25 pence and one order addition of 3,300 stock units at price level 313.75 pence. In Figure 7, the difference between the LOB at 10:27:06 GMT and 10:27:07 GMT reflects these order activities.

**CONCLUSION**

The reconstruction of the LOB of electronic order-driven systems from UHF data has become an active area for research. The challenge is to find a programming solution that can achieve an accurate and extremely fast reconstruction of the LOB. Since each event that occurs in the LOB is recorded, time stamped and given a unique ID and the rules of order matching (that is, price and time priority) are clear, a hash table lends itself as a natural programming solution to LOB reconstruction. In this paper, we demonstrated the application of SAS® Hash Object to the reconstruction of SETS electronic LOB of the LSE.
REFERENCES


ACKNOWLEDGMENTS

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies.

I would also like to take this opportunity to thank the various users and contributors to Google Online SAS Users’ Forum in particular, to name but a few, Messrs. Paul Dorfman, Howard Schreier, Richard A. DeVenezia and Chang Y. Chung. I have benefitted immensely from their solutions to the SAS coding challenges posted by SAS users over the years. https://groups.google.com/group/comp.soft-sys.sas/topics?lnk=srg&hl=en. Heart warm thanks to my PhD supervisor, Alfonso Dufour, for introducing me to SAS many years ago.

I am grateful for comments from Ryan Davies (Babson College), Mark Keintz (Wharton Research Data Services) and Alfonso Dufour (ICMA Centre, Henley Business School). All errors or omissions are my responsibility.

CONTACT INFORMATION

For Comments and suggestions, please contact the author at:

Dr. Dorian M. Noel, CAIA  
Department of Management Studies  
The University of the West Indies, St. Augustine  
Trinidad and Tobago  
Tel. (868) 662 2002 ext. 2105  
Fax: (868) 662 1140  
E-mail: dorian_noel@sta.uwi.edu  
Website: http://sta.uwi.edu/fss/dms/

Dr. Dorian M. Noel is a former Commonwealth Scholar to the International Capital Market Association Centre (The Business School for Financial Markets) at Henley Business School, University of Reading, where he obtained his PhD in Finance. Dr. Noel has worked in the fields of quantitative finance, risk and portfolio management, energy economics and economic policy formulation. He is currently a Lecturer in Finance at the University of the West Indies, St. Augustine and teaches at both the postgraduate and undergraduate levels in the areas of risk management, financial econometrics, derivatives, portfolio management and corporate finance. His current research interests include empirical microstructure (algorithmic trading and limit order book modelling), portfolio and risk management, alternative investments and financial econometrics. Dr. Noel is also a designated Chartered Alternative Investment Analyst (CAIA) and a member of Global Association of Risk Professionals (GARP).