Momentum Strategy Module Specification

Version:	1.0
Date Modified:	03 Mar 2014

1. The context

We are a company specialising in high frequency algorithmic trading on financial markets. We have an existing system with predefined trading strategies. We wish to open our system to third parties software houses, offering them ability to "plug-in" independent software modules that implement particular strategies. We wish to conduct an evaluation as to what is the best approach to integrate independent software modules with our existing system without each party revealing their source code. We therefore request all interested companies to provide an independent software module that implements a well-known trading strategy called "Momentum". Besides knowing which team can do it best, we would like to know what is their suggested approach for integrating the module with our system.

Figure 1 shows how our system interacts with the Module Strategy Module (MSM).

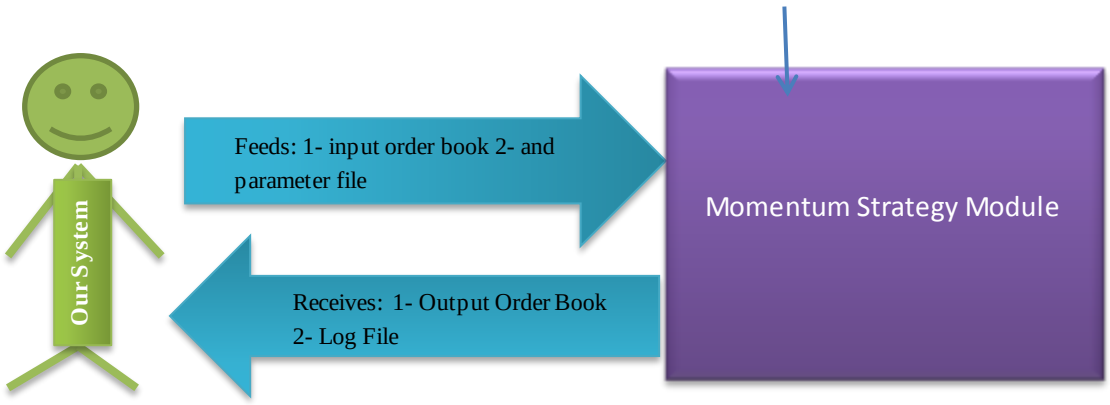

Figure 1High level trading platform architecture

The language in which the module is to be written is not important as long as there is some way to invoke it from our system. Also, all exchanged data is in the form of text files to avoid any specific encoding. In other words, our system calls the module and supplies any data in the form of text files. After execution is finished, all results are also in text files. It is important that the module can be used without its source code being revealed as the Intellectual Property associated with the trading strategy will stay with the software house.

2. Module Specification

Description of the trading strategy

At this stage, it is not important to know exactly what a trading strategy is. For now, we will just describe the way the module works in terms of data processing steps which can be understood by anyone with limited knowledge about financial trading. These various data transformation steps are shown in Figure 2.

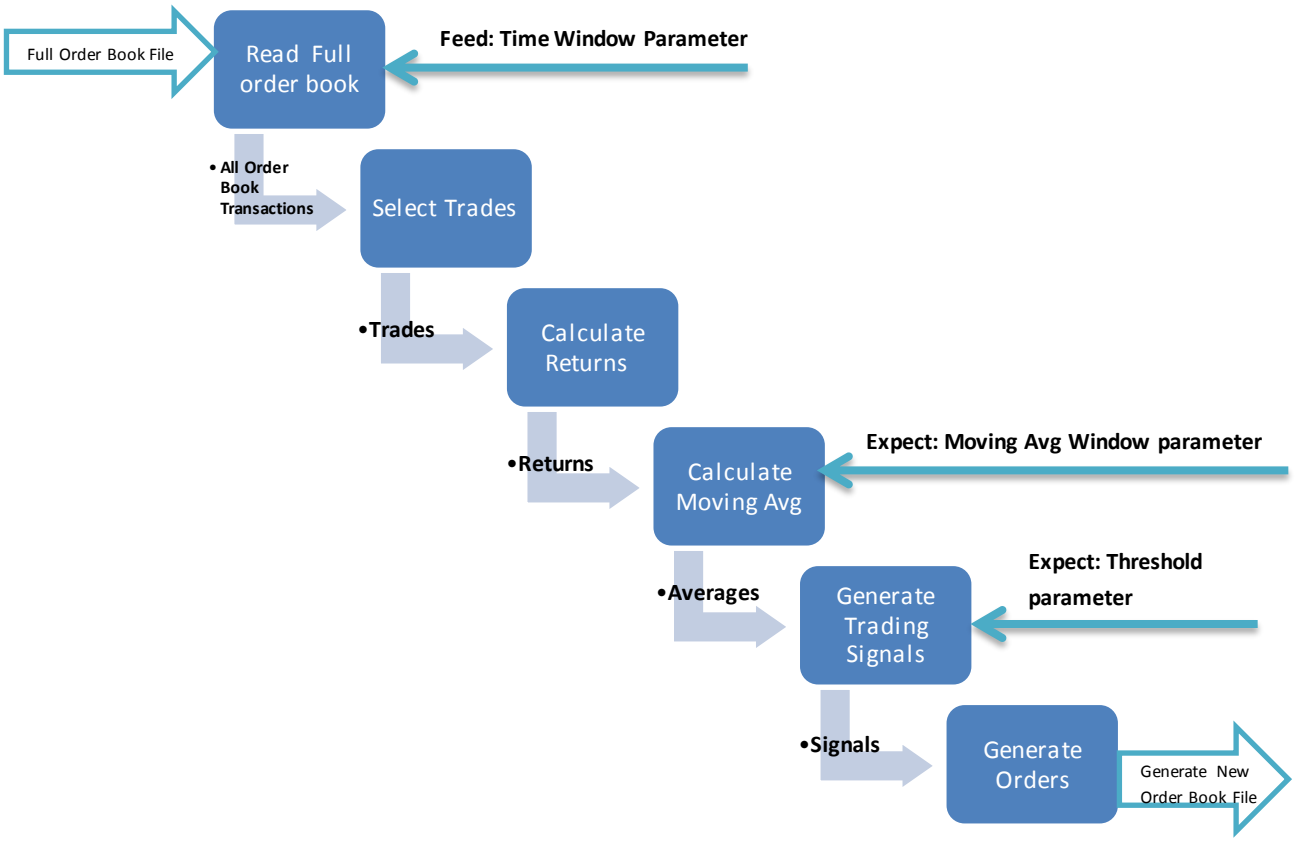

Figure 2 MSM data processing steps

Step1: Select trades

This step reads a "Full Order Book" (in CSV format) as input. Figure 3 shows a sample file opened in Excel. The purpose of this step is to select lines of type "TRADE" some of which are highlighted in **Green** in figure 3.

- **Input**: Full order book file as shown in figure 3, the file format is CSV.
- **Processing**: Select only "Record Type" of type TRADE
- **Expected output:** List of Trades records only**.**

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	#Instrument	Date	Time	Record Type	Price	Volume	Undisclosed Volume	Value	Qualifiers	Trans ID	Bid ID	Ask ID	Bid/Ask	Entry Time	Old Price	Old Volume	Buyer Broker ID	Seller Broker ID
2	BHP	20130205	12:17:18 PM	AMEND	37.24	196		7299.04		0		6230424149652770000.00	A	12:17:08	37.24	47		210
3	BHP	20130205	12:17:19 PM	ENTER	37.23	1059	0	39426.57		0	6230424149652770000.00		B				361	
4	BHP	20130205	12:17:19 PM	AMEND	37.26	2		74.52		0		6230424149652720000.00	A	12:09:30	37.27	2		110
5	BHP	20130205	12:17:19 PM	AMEND	37.27	2		74.54		0		6230424149652720000.00	A	12:09:30	37.26	2		110
6	BHP	20130205	12:17:19 PM	ENTER	37.23	557	0	20737.11	OB0	0		6230424149652770000.00	A					298
7	BHP	20130205	12:17:19 PM	TRADE	37.23	190		7073.7		0	6230424149652760000.00	6230424149652770000.00					299	298
8	BHP	20130205	12:17:19 PM	TRADE	37.23	4		148.92		0	6230424149652760000.00	6230424149652770000.00					150	298
9	BHP	20130205	12:17:19 PM	TRADE	37.23	24		893.52		0	6230424149652760000.00	6230424149652770000.00					297	298
10	BHP	20130205	12:17:19 PM	TRADE	37.23	130		4839.9		0	6230424149652760000.00	6230424149652770000.00					217	298
11	BHP	20130205	12:17:19 PM	TRADE	37.23	209		7781.07		0	6230424149652760000.00	6230424149652770000.00					110	298
12	BHP	20130205	12:17:19 PM	ENTER	37.23	360	0	13402.8		0	6230424149652770000.00		B				299	

Figure 3 Full order book downloaded from Aus Equities

At this stage, no need to fully understand this file but you can find details in AusEquities Order Book File Format . Such file can be downloaded from AUS Equities website [1]. You will be given an account and instructions on how to access the system to retrieve your own CSV files for testing purposes.

Step2: Calculate returns (Rate of Change)

Each trade selected in the previous step will have a price associated with it and timestamp. The price is referred to it as $\boldsymbol{P_t}$, which is the price at time $\boldsymbol{t}$. Column E in Figure3 above represents $\boldsymbol{P_t}$ values and column C represents the $\boldsymbol{t}$ values.

Returns are calculated according to the following equation:

$$R_t = (P_t - P_{t-1})/P_{t-1}$$

Where Pt = price traded at time t and P_{t-1} is the price traded as time $(t-1)$ and R_t is the return at time t .

Example 1

Table 1 shows an example of a series of trade prices and the corresponding returns.

Table 1 Calculating Returns

t	1	2	3	4	5	6	7	8	9
P_t	$30	$30.34	$30.71	$31.09	$31.29	$31.54	$31.76	$31.98	$32.58
Rt		0.011	0.012	0.012	0.0064	0.0079	0.0069	0.0075	0.0187

So in summary, this step works as follows:

- **Input**: List of trades generated by the previous step
- **Processing**: computing returns according to the above formula
- **Expected output**: list of returns

Step3: Calculate Simple Moving Average (SMA)

This step takes a number (*n>1*) as a parameter and computes simple moving averages over a window of *n* returns. A simple moving average at time *t*, referred to as SMA_t, is calculated using the following equation:

$$SMA_t = \frac{\sum_{m=(t-n)+1}^{t} R_t}{n} \qquad (t \geq n)$$

Where R_t is the return at time t

Example 2

If we assume $n = 3$, by applying the $\boldsymbol{SMA_t}$ equation to the returns in Table 1, we have the results shown in Table 2

Table 2 Calculating Simple Moving Avg with n=3

t	3	4	5	6	7	8	9
m	$\{t1, t2, t3\}$	$\{t2, t3, t4\}$	$\{t3, t4, t5\}$	$\{t4, t5, t6\}$	$\{t5, t6, t7\}$	$\{t6, t7, t8\}$	$\{t7, t8, t9\}$
Rt	0.012	0.012	0.0064	0.0079	0.0069	0.0075	0.0187
$\boldsymbol{SMA_t}$	0.0076	0.0116	0.0101	0.0087	0.0070	0.0074	0.0110

If n was 4 then the first three values of $i\ and\ m\ will\ be\ the\ same$ and the values of the moving average are as shown in Table 3.

Table 3 Calculating Simple Avg with n=4

t	4	5	6	7	8	9
m	$\{t1, t2, t3, t4\}$	$\{t2, t3, t4, t5\}$	$\{t3, t4, t5, t6\}$	$\{t4, t5, t6, t7\}$	$\{t5, t6, t7, t8\}$	$\{t6, t7, t8, t9\}$
Rt	0.012	0.0064	0.0079	0.0069	0.0075	0.0187
$\boldsymbol{SMA_t}$	0.00875	0.01035	0.009575	0.0083	0.007175	0.01025

Figure 4 shows a comparison between the return and moving average values.

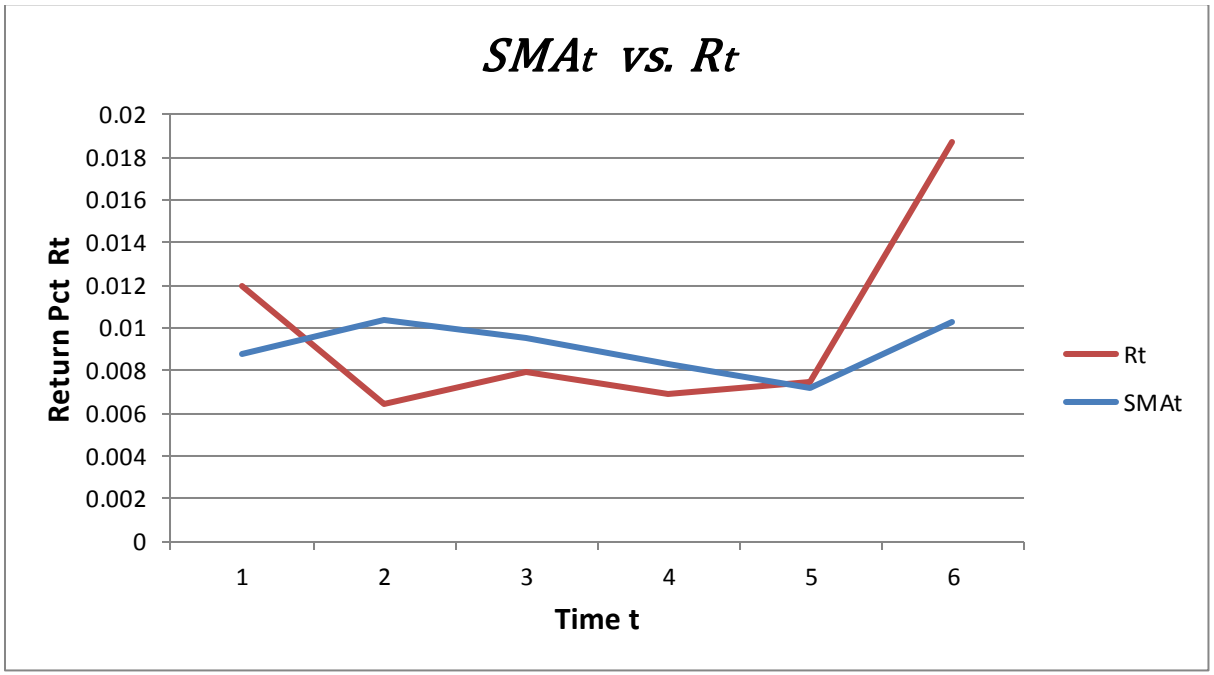

Figure 4 Simple Moving Avg vs. Returns

So in summary, this step works as follows:

- **Input**: List of returns
- **Parameter**: Moving Avg Window parameter n will determine the number of returns to be included in the calculation of Moving Avg values.
- **Processing**: Using the moving average formula
- **Expected output**: list of moving averages values.

Step4: Generate Trading Signals with Threshold

This step converts moving averages calculated in the previous step into trading signals. The two signals generated are either **BUY** or **SELL** signals. The momentum strategy first computes successive differences between moving average values as follows:

$$TSv_t = SMA_t \ - \ SMA_{t-1}$$

Where TSv_t is the Trading Signal value, SMA_t is the Simple Moving average at time t, and SMA_{t-1} is the simple moving average at time $t-1$.

If the value of TSv_t is above a certain threshold TH, the strategy would generate a buy signal, if it's lower than the threshold, it generates a sell signal. Therefore, the rule applied to generate the trading signals is as follows:

$$TS_t = \begin{cases} \mathrm{BS}_t\ , if\ TSv_t > \mathrm{TH} \\ SS_t\ , if\ TSv_t < -\mathrm{TH} \\ Not\ Defined, Otherwise \end{cases}$$

Where BS_t Buy Signal is at time t, SS_t is the Sell Signal at time t. The threshold value TH is passed as a parameter to the MSM module.

Example 3

If TH was 0.001 then the TSv_t and TS_t from Table 3 will be as shown in Table 4 below

i	4	5	6	7	8	9
m	$\{t1, t2, t3, t4\}$	$\{t2, t3, t4, t5\}$	$\{t3, t4, t5, t6\}$	$\{t4, t5, t6, t7\}$	$\{t5, t6, t7, t8\}$	$\{t6, t7, t8, t9\}$
Rt	0.012	0.0064	0.0079	0.0069	0.0075	0.0187
$\boldsymbol{SMA_t}$	0.00875	0.01035	0.009575	0.0083	0.007175	0.01025
TSv_t	Undefined	0.0016	-0.000775	-0.001275	-0.001125	0.003075
TS	Undefined	Buy Signal	Nothing	Sell Signal	Sell Signal	Buy Signal

Step5: Generate orders

This step generates buying or selling orders using the signals generated from the previous step. A buy order is only generated when the first Buy signal is encountered and a Sell order is only generated when the first Sell signal is encountered.

Example

If the Trading signals generated from the previous step were as follows:
BS1,BS2,BS3,SS1,SS2,SS3,SS4,BS4

Where BS1 = Buy Signal 1 , SS1 = Sell Signal 1 and so forth

The orders generated should be as follows:

Buy order for BS1 order

Sell order for SS1

Buy order for BS4

So in this case, only three entries are generated, ignoring consecutive similar signals, and only generating orders when the direction of the signal changes.

Buy and sell orders generated by this step must be stored in a CSV file with a format similar to the one of the input file. Figure 3 shows examples of orders which correspond to "ENTER" transactions:

- Line 3 is a buy order because it has the letter "B" in the column Buy/Sell
- Line 6 is a sell order because it has the letter "A" in the column Buy/Sell

The other fields must be completed as follows:

- Date/time: must correspond to the date and time of the corresponding P_t (from Step 1) which generated this signal
- Price: must correspond to the value the corresponding P_t (from Step 1) which generated this signal
- Volume: not important, use a fixed value like 100
- Value: is price multiplied by the volume
- BidID/AskID: you can generate any numbers as long as they are not identical
- Other columns can be left empty

Exchanging data with the module

As mentioned earlier, all data exchanged with the module is via text files. There are two input files:

- A full order book data file: must be formatted according to the Sirca format. The module should be able to process files downloaded from the Sirca web site without any modifications made to them
- A parameters file: you are free to choose any format you like as long as the way these parameters are passed from our system is fully documented

There are two output files:

- An orders data file: these are the orders generated by the module. They must be formatted according to the Sirca format (ENTER transactions).
- A log file: you are free to choose any format you like as long as the information specified below is contained in it

The log file must contain the following information:

- Developer team
- Module name and version
- Input file name
- Parameters passed like (number of days, Trading threshold, window, instrument studied …etc.)
- An indication if execution has been successful or there is an error
- If error, indicate the nature of the error
- If successful, need to supply
    - Start date and time of execution
    - End date and time of execution
    - Elapsed time
    - Output file name

3. Additional Information:

Teams have the choice of running their system on two different platforms:

- PC running Windows
- Unix/Linux platform

Throughout the workshop, each team will need to have a Web page. As a minimum, the page is showing:

- The team name and members
- Consecutive releases of their module. Each release page must include a link to download the module and information about:
    - The date and version of the release
    - What has been implemented so far
    - Differences with previous version
    - Clear instructions on how to run the module in standalone mode
    - Guidelines on how to integrate the module with other systems
    - Any test software or data

4. References:

1. Aus Equities Order Book File format
   http://www.cse.unsw.edu.au/~fethir/TradingEngineCourse/Session1/section%201%20links%20and%20topic/Revised%20introOrderbook-1-3.htm
2. Momentum Trading Strategy
   http://www.onlinetradingconcepts.com/TechnicalAnalysis/Momentum.html