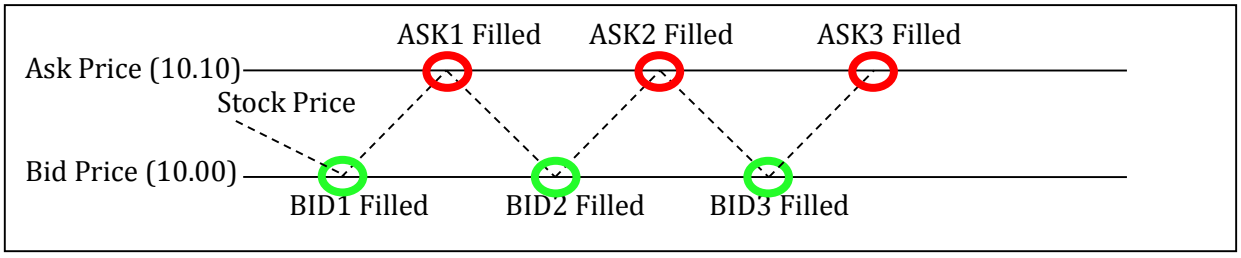
Algorithmic Market Making

This case is designed to build on skills learned in the Algorithmic Arbitrage (ALGO1) case and motivate students to build a market-making algorithm that generates profits by capturing the bid-ask spread.

A market making algorithm is considerably more difficult to conceptualize since trading occurs over time (versus arbitrage where all trades occur simultaneously). Capturing a bid-ask spread will require traders to inventory a long (or short) position for an uncertain period of time as they wait for an opposing trade to cover their position.

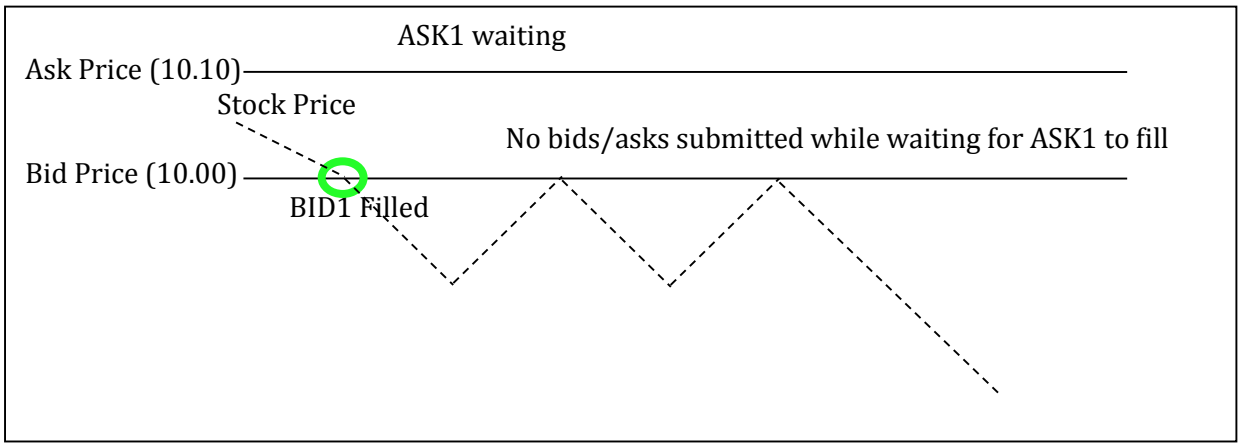
The simplest execution of a market-making trade is to submit a paired bid and offer and have the two orders filled over time; the trader (algorithm) earns the price differential. When markets aren't trending, this is a reasonably effective strategy.



The trader submits their first "pair" of orders (BID1 and ASK1) of equal size and waits for the two to become filled. The price trades down to the bid price, which fills, and the trader has an inventory of shares. The trader then waits, and the price trades up to the ask price and they receive a fill at the ask price, extinguishing their inventory. They no longer have a position, and they have successfully captured a 10 cent bid/ask spread. The trader can then submit a new pair of orders and repeat the process.

Kevin Mak and Tom McCurdy** prepared this case for the RIT market simulation platform, http://rit.rotman.utoronto.ca/.*
**Manager of the Financial Research and Trading Lab, Rotman School of Management;*
***Professor of Finance and Founding Director of the FRTL, Rotman School of Management, University of Toronto.*

This logic fails in a market that does not stay in a small trading range, because if one of the orders does not get filled, the algorithm waits (potentially) a very long time, and fails to capture the bid ask spread during that time.



The logical answer to this is that if a pair order does not get filled on both sides (that is, either the bid or the ask order does not get filled), one would want to simply "reset" the algorithm and start submitting new sets of pairs. From an algorithmic perspective, we can simplify this into two simple rules:

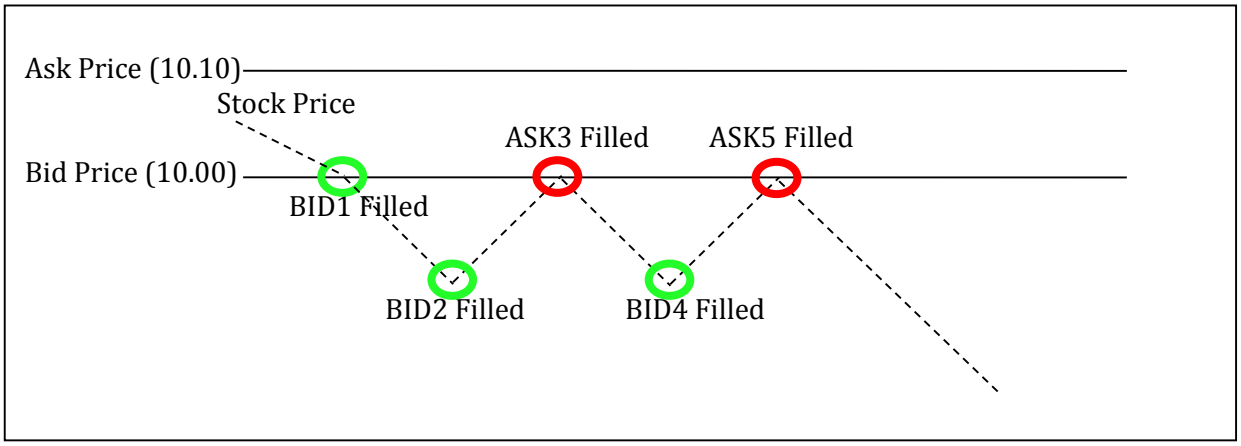
1. We always want to have a bid and an ask order in the market. For example, the bid could be submitted at (LAST – Spread) and the ask submitted at (LAST + Spread). We want our pair of orders to be close to the top of the book.
2. We want our position to be close-to-balanced whenever possible. When markets are trending and we're getting filled on more bids than asks, we want to adjust our algorithm accordingly*.

Programmatically, your algorithm should do the following:

1. Constantly check to see if you have any orders in the order book. If you do not, then you should submit your bid and ask. If you only have one order in the order book, cancel it.
2. Check your inventory and adjust your bid/ask prices or quantities to try to balance your inventory.

This logic will accomplish the following:

1. BID1 and ASK1 are submitted
2. BID1 is filled; ASK1 remains unfilled so it will be cancelled.
3. BID2 and ASK2 are submitted.
4. BID2 gets filled, ASK2 is cancelled
5. BID3 and ASK3 are submitted.
6. ASK3 is filled, BID3 will be cancelled



Using your skills from ALGO1, try to program an algorithm that will accomplish this procedure. If you are having difficulties, consult the help file "VBA API Documentation".

*Think about how you would adjust your algorithm "accordingly" to execute the trades and attempt to balance your inventory.

Algorithmic Trading Simulation #2 – ALGO2

In the ALGO2 case, there will be one stock (ALGO) available to be traded.

There is a limit of 5,000 shares per order, and each student is limited to a position of 25,000 shares gross or net. There is a 1 cent per share trading commission charged on all market orders (active liquidity). There is a ½ cent per share rebate provided to all limit orders (passive liquidity) that are filled. Trading will take place over 5 minutes (300 seconds). There is a fine of 10 cents per share charged whenever a trader exceeds their position limit of 25,000 shares. For the ALGO2 case, the Rotman Application Programming Interface (API) is enabled and can be accessed in Microsoft Excel in the following manner:

1. Turn on the Developer ribbon
2. In the Visual Basic window, go to "Tools → References" and add "Rotman Interactive Trader"
3. In any function or subroutine, use the following two lines of code to initialize the API:
   Dim API As RIT2.API
   Set API = New RIT2.API

4. A full list of API commands can then be accessed using "API.<*Insertcommand*>" in VBA – for example, "API.AddOrder()"

Discussion Questions and Follow Up:

(1) How would you decide what the appropriate bid/ask spread should be when you submit your pair orders? Should this be dynamic or static?

(2) What are the different ways you can alter your trading strategy, so that you keep submitting orders but attempt to balance your book?

(3) Will market making strategies typically work in a trending market?

(4) What is the implication of executing this strategy on a large number of correlated securities?

(5) How do trading rebates (instead of commissions) alter the economics of market making? Consider situations where a market maker uses limit orders (passive liquidity) and makes trades where they have a profit of -1 cent, zero cents, and +1 cent excluding fees/rebates.