ORF 445 High Frequency Markets
Final Homework

Robert Almgren

April 26, 2021
due 5 PM Wednesday, May 5 (Dean's Date)

Choose and turn in **one** of the following three problems.

1. Multidimensional Cointegration

   In Homework 4 you implemented a cointegration model between two assets. For product complexes in which many related products are all active (Eurodollars, for example), you can get better performance by including more of those assets. This is essentially an n-dimensional version of Homework 4. It is a bit more complex, but should give better results.

   Choose a set of 8 closely related products. I suggest Eurodollars maturing Mar 2021 through Dec 2022: `GEH1`, `GEM1`, `GEU1`, `GEZ1`, `GEH2`, `GEM2`, `GEU2`, and `GEZ2`. (The near maturity Dec 2020 `GEZ0` may have strange behavior.) You may also try the 8 most active Crude Oil maturities. Sample the quote midpoint prices at 1-minute intervals during the entire 23-hour trading day, for any one trading day in October 2020.

   The assumption is that all these 8 prices pretty much move back and forth along a 1-dimensional line. You can take more degrees of freedom (for example, they move around in a 2-plane in $\mathbb{R}^8$), but one d.o.f. should give pretty good results.

   Compute the rolling average mean prices and the rolling average covariance matrix just as for the 2-dimensional example in the homework. In the homework, we took sums all the way back to the beginning of the trading day for each time point. But since now you want to do this in a more realistic way, use exponential averages as discussed in class (and I will post some notes). Use say a 1-hour averaging time.

   At each time, compute the singular value decomposition of the covariance matrix, and compute a vector forecast price P_*, the projection of the actual price P onto the first principal component of this covariance matrix.

   For each product, the signal S is the corresponding component of $P_* - P$. Evaluate this signal by taking its correlation with forward price changes, for one maturity in the middle of the range (`GEZ1` or `GEH2`). If you get this working right, then you should get somewhat better forecasting ability than for the 2-dimensional example in the homework.

2. Trade Simulation

   A common way that execution algorithms work is to place limit orders on the appropriate side of the market (bid for a buy order, ask for a sell order). Then for each limit order, wait to see if it gets a passive fill which will capture the bid-ask spread. But if signals indicate that the price is likely to move away, then cancel your limit order and cross the spread to execute before the move happens. This problem shows one way in which you might simulate and evaluate such a crossing strategy.

   Choose some futures contract from our data set, preferably a reasonably large-tick one whose bid-ask spread is nearly always one price increment. Choose a microprice threshold value ρ: for example, $\rho = 85\%$ means you will cross when the microprice is 0.85 of the way to the far side. For a buy order, this means buy on the ask when $q_{\mathrm{B}}/(q_{\mathrm{A}} + q_{\mathrm{B}}) > \rho$; for a sell order this means sell on the bid when $q_{\mathrm{A}}/(q_{\mathrm{A}} + q_{\mathrm{B}}) > \rho$.

   Here is how you can test the value of this signal, and evaluate different values of ρ:

   (a) Choose a random date, and a random time during the trading day between 07:00 and 15:00. Choose a random side: buy or sell. I describe the process for a buy; the process for a sell is the same but reversed.

   (b) Imagine placing a limit order for a single lot on the bid. Record the bid size when you place the order, and the midpoint price.

   (c) Step forward through quote and trade data in sequence forward from that time (you may like to use the function `tqmerge`, described in the data writeup) until one of these three things happens:

      i. Trade volume at the bid price with aggressiveness `S` prints that is greater than or equal to the bid size when you placed the order, or the bid price decreases below the price at which you placed the order. In this case, your trade executes at its resting price. Or

      ii. The microprice moves above ρ. Then cross the spread to buy at the ask. Or

      iii. The bid moves up. Then move your quote up to the new bid and repeat until one of the first two events happens. (In practice, you would wait a bit of time to see whether the quote will flick back before you move your order.)

   (d) The slippage of this strategy is the final execution price minus the bid-ask midpoint at the time you placed the order (with sign reversed for a sell). Repeat a large number of times (at least 100) with random dates, times, and sides, to measure the average slippage for this strategy. This gives a reasonable estimate for how well an execution algorithm would work.

   Repeat this for several values of ρ between 0.5 and 1, and see which value gives the best slippage.

3. Technical Indicators

   This is to test whether any of the technical indicators from Lecture 10b[1] have any value. For this, you will want to return to the WRDS data service, but this time rather than the TAQ dataset for tick data, use the CRSP data for daily stock prices.

   Pick a stock that is not too large nor too small (say more than 100 but less than 1000 ranking by market cap. Download at least two years of daily data from CRSP via WRDS. You want at least the fields `date` for date, `open` for opening price, and `close` for closing price. You also will need `cfacpr` to correct prices for dividends (read the documentation to understand exactly how to use this, or find a stock for which `cfacpr` is always 1). Test the technical indicators listed below.

   For each of these, assume that at the open of day D, you have computed the indicator based on closing prices on days $D-1, D-2, \dots$, and possibly the open on day D. The result of the indicator will be either "buy," "sell," or no signal. If you receive a "buy" signal based on the close prices through day $D-1$, then you buy a fixed amount on the open of day D and sell on the close of day D; if you receive a "sell" signal based on close prices through day $D-1$, then you short on the open of day D and repurchase on the close price of day D. Thus, in effect, you are looking for a correlation between your signal and the price change from open to close on day D. (You lose the possible overnight return you would get by opening the position at the close of day $D-1$, since it takes you some time to compute the signal.)

   - Channel breakout (Lecture 10b slide 58). Compute the backward-looking maximum and minimum over the previous k days: $M_D = \max\{P_{D-1}, \dots, P_{D-k}\}$ and $m_D = \min\{P_{D-1}, \dots, P_{D-k}\}$ and set say $k = 10$ days. Here P_{D-j} is the close price on day $D-j$. Set a threshold δ (you will have to try several values). Generate a "buy" signal on day D if $P_D^{\text{open}} > M_D + \delta$, or a "sell" signal if $P_D^{\text{open}} < m_D - \delta$. This assumes you can execute the trade at the open of day D.
   - Moving average crossover (Lecture 10b slide 59). Compute two different exponential moving averages (as in Notes 9b page 4) on each day D: $\overline{P}_D^{\text{fast}}$ with a time scale $t_{\text{fast}} = 5$ days and $\overline{P}_D^{\text{slow}}$ with a time scale $t_{\text{fast}} = 40$ days. Generate a "buy" signal on day D if $\overline{P}_D^{\text{fast}} > \overline{P}_D^{\text{slow}} + \delta$, or a "sell" signal on day D if $\overline{P}_D^{\text{fast}} < \overline{P}_D^{\text{slow}} - \delta$.
   - Head and Shoulders and inverse (Lecture 10b slide 65). Compute moving averages $\overline{P}_D$ as above, but now use a much shorter averaging time, say 2 days. Track the sequence of local maxima and minima of $\overline{P}_D$ (the exponential averaging is only to smooth out fluctuations and not generate too many local maxima and minima). Generate a "sell" signal when a head-and-shoulders pattern is seen or a "buy" signal when the inverse is seen.

   Evaluate whether the signal yields a positive profit at the end of your trading period.

[1] Or look up the paper by Lo, Mamaysky, and Wang—I can put it on Canvas if people want.

4. Optimal Trajectories with Bayesian drift

   Suppose that, just as discussed in class and in the homework, you must buy X shares of some stock between times $t = 0$ and $t = T$. Market impact is as discussed in class, with a linear price impact function $g(v) = \gamma v$ where $v(t)$ is the rate of buying. Assume permanent impact is zero for this problem. But in this problem you are *risk-neutral:* you do not care about the variance of the final cost. You only want to minimize the expected value of the total cost.

   As in the homework, there is a constant drift α, but the big difference is that you do not know the value of α. However, you can begin to estimate it as you observe the price motion early in the day, and you can use that estimation to improve your execution through the remainder of the day. You do know the the volatility σ.

   The intuition is this: Suppose you observe the price moving up at the beginning of the day. This could be a random fluctuation, or it could be the beginning of an persistent upward drift. If the price will continue to move up, then you would like to accelerate your buy program to buy before it goes up. But because of market impact, buying faster is expensive, so you do not want to accelerate too much until you have observed enough data to have some degree of certainty about the direction.

   Specifically, at time t, knowing the price $P(t)$ and the initial price $P(0)$, your best estimate for the drift at that time is (you can derive this using Bayesian statistics)

$$\alpha_*(t; P(t)) = \frac{P(t) - P(0)}{t}.$$

   You know that this value will change in the future as you observe future values of $P(t)$, but you do not know how nor in which direction it will change. One way to construct trajectories is to *assume* that α will remain constant for $t \le s \le T$ at its value $\alpha = \alpha_*(t; P(t))$, and determine an optimal trajectory $x(s)$ and $v(s)$ for $t \le s \le T$ using this assumed constant value. Of course, a short time Δt later, when $P(t)$ has changed to $P(t + \Delta t)$ and you have computed a new value of α_*, you will change the trajectory. But at least this will give you the trade rate $v(t)$ at time t, in terms of t and $P(t)$.

   Write the optimal trade rate $v(t, P(t))$ generated by the above procedure. This is a simple example of an adaptive execution program, that adjusts its trade rate depending prices observed during the trade execution.

   Two additional comments:

   - Trajectories generated by the above procedure may temporarily exceed the target level or go below zero. That is, if $\alpha > 0$, it may be optimal to buy *more* than X temporarily then sell it back before time T. If $\alpha < 0$, then it may be optimal to short the stock (assuming that is possible), then buy back the short position as well as the target X before time T. In real practice, you would probably need to impose the constraint $0 \le x(t) \le X$, so that trading never reverses sign, but for this problem ignore that constraint.
   - One can show that the above procedure actually does generate the real optimal trajectory, despite the seemingly ridiculous assumption that α will not change although you know that it will.