MMAT Homework 5

Pittsburgh Campus
Jingyi Guo (jingyig1), Jiawen Zhang(jiawenz2)

1. Market Making Strategy

We implement the market making strategy for ZNU7 on Aug. 23.
Code:

```
from qpython import qconnection
import numpy  as np
import pandas as pd
import matplotlib.pyplot as plt
import datetime

def qSym(sym):
    return "`" + sym

def tsfmt(x):
    if x >= 0:
        return '0D%02d:00:00'%(x)
    else:
        return '-0D%02d:00:00'%(-x)

class MarketMakingStrategy(object):
    def __init__(self, db, d, sym, tL=-7, tR=16, maxPos=100, nmax = 0):
        self.db = db
        self.d = d
        self.sym = sym
        self.tL = tL
        self.tR = tR
        self.maxPos  = maxPos
        self.nmax    = nmax

    def queryData(self):
        query = "select from tqmergeT[" + self.d.strftime("%Y.%m.%d") + ";" +
qSym(self.sym) +";" + tsfmt(self.tL) + ';' + tsfmt(self.tR) + "] where (differ
bid)|(differ ask)"
        print(query)
        self.marketData  = self.db(query)
        query = 'select minpxincr, dispfactor, notional from instinfo where inst
=sym2inst[' + qSym(self.sym) +']'
        print(query)
```

```
        contractSpec = self.db(query)
        self.pTick = contractSpec.minpxincr[0]
        self.dispfactor = contractSpec.dispfactor[0]
        self.notional = contractSpec.notional[0]
        self.x = 0
        self.P = 0
        self.M = 0.5 * (self.marketData.bid[0] + self.marketData.ask[0])
        self.bPrice = np.nan
        self.aPrice = np.nan
        self.bSize = np.nan
        self.aSize = np.nan
        self.tradeTime    = [min(self.marketData.t)]
        self.curPos     = [self.x]
        self.curCashFlow   = [self.P]
        self.curMidPrice   = [self.M]

    def trade(self, tseq, tTime, ssiz, mid, prc):
        if pd.isnull(prc): # quote
            if ssiz > 0:
                prc = self.bPrice
                self.bPrice = np.nan
            else:
                prc = self.aPrice
                self.aPrice = np.nan
            self.x += ssiz
            self.P -= ssiz * prc
            self.tradeTime.append(tTime)
            self.curPos.append(self.x)
            self.curCashFlow.append(self.P)
            self.curMidPrice.append(mid)

    def runStrategy(self):
        bPrev, bsPrev, aPrev, asPrev = np.nan, np.nan, np.nan, np.nan
        for curdata in self.marketData.itertuples():
            if pd.isnull(curdata.prc):
                pmid = 0.5 * (curdata.bid + curdata.ask)

                if not pd.isnull(self.bPrice):
                    if curdata.bid < self.bPrice:
                        self.trade(curdata.seq, curdata.t, +1, pmid, np.nan)
                    elif curdata.bid > self.bPrice:
                        self.bPrice = np.nan
```

```
                if not pd.isnull(self.aPrice):
                    if curdata.ask > self.aPrice:
                        self.trade(curdata.seq, curdata.t, -1, pmid, np.nan)
                    elif curdata.ask < self.aPrice:
                        self.aPrice = np.nan
                    bPrev, bsPrev, aPrev, asPrev = curdata.bid, curdata.bsiz,
curdata.ask,curdata.asiz

            else:
                pmid = 0.5 * (bPrev + aPrev)

                if not pd.isnull(self.bPrice):
                    if curdata.prc < self.bPrice:
                        self.trade(curdata.seq, curdata.t, +1, pmid, np.nan)
                    elif curdata.prc  == self.bPrice:
                        self.bSize = self.bSize - curdata.siz
                    if self.bSize < 0:
                        self.trade(curdata.seq, curdata.t, +1, pmid, np.nan)

                if not pd.isnull(self.aPrice):
                    if curdata.prc > self.aPrice:
                        self.trade(curdata.seq, curdata.t, -1, pmid, np.nan)
                    elif curdata.prc  == self.aPrice:
                        self.aSize = self.aSize - curdata.siz
                    if self.aSize < 0:
                        self.trade(curdata.seq, curdata.t, -1, pmid, np.nan)

            if pd.isnull(self.bPrice) and self.x < self.maxPos:
                self.bPrice, self.bSize = bPrev, bsPrev

            if pd.isnull(self.aPrice) and self.x > - self.maxPos:
                self.aPrice, self.aSize = aPrev, asPrev

            self.trade(max(self.marketData.seq), max(self.marketData.t), -
self.x,bPrev if self.x > 0 else aPrev, 0.5 * (bPrev + aPrev))
            self.res = pd.DataFrame({'tradeTime': self.tradeTime, 'position':
self.curPos, 'cashFlow': self.curCashFlow, 'midPrice': self.curMidPrice})
            self.res['PnL'] = self.res.cashFlow + self.res.position *
self.res.midPrice
    def plot(self):
        plt.figure(figsize = (8,6))
```

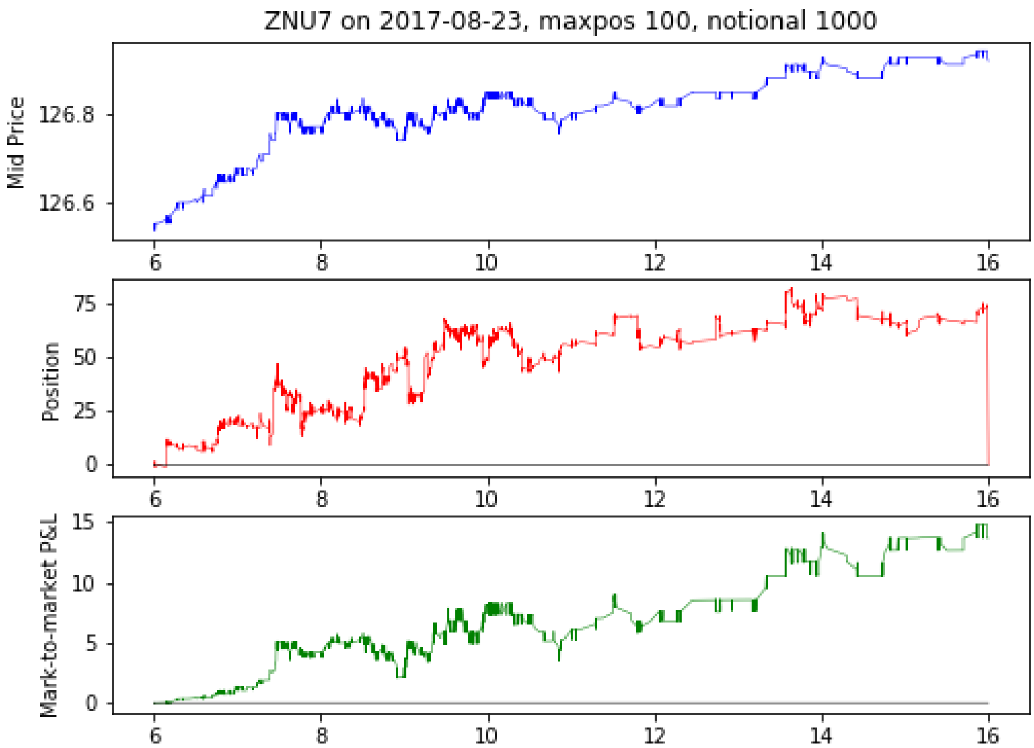
```
        plt.subplot(311)
        plt.plot(self.res.tradeTime, self.res.midPrice, linewidth=.5, color='b')
        plt.ylabel('Mid Price')
        plt.title('{} on {}, maxpos {}, notional {:.0f}'.format(self.sym,
self.d,self.maxPos, self.notional))
        plt.subplot(312)
        plt.plot(self.res.tradeTime, self.res.position, linewidth=.5, color='r')
        plt.plot(self.res.tradeTime, np.zeros(self.res.tradeTime.shape), linewidth
=.5, color='k')
        plt.ylabel('Position')
        plt.subplot(313)
        plt.plot(self.res.tradeTime, self.res.PnL, linewidth=.5, color='g')
        plt.plot(self.res.tradeTime, np.zeros(self.res.tradeTime.shape), linewidth
=.5, color='k')
        plt.ylabel('Mark-to-market P&L')
        plt.savefig('MMResult.png')
        plt.show()

def main():
    curDT = datetime.datetime.now()
    db_IR = qconnection.QConnection(host='kx', port=6000, pandas = True)

    # initialize connection
    db_IR.open()
    d = datetime.date(2017, 8, 23)
    sym = 'ZNU7'
    tL =6
    tR = 16
    maxPos = 100
    nmax =0
    mms = MarketMakingStrategy(db_IR, d, sym, tL, tR, maxPos, nmax)
    mms.queryData()
    mms.runStrategy()
    mms.plot()
    print(datetime.datetime.now() - curDT)

if __name__ == '__main__':
    main()
```

Output:



- Reasons why this strategy might be profitable or not profitable:

  According to the P&L plot, this strategy is generally profitable on Aug 23, this might because that market making strategy is generally profitable on mean reverting time series, and the ZNU7 prices somehow follows a mean-reversion pattern on Aug 23.

- How the strategy could be improved:

  We can take into consideration the bid/ask size in the limit order book when buliding our inventory.

  In addition, we could try dynamic inventory management instead of fixed constraints.

2: Given ① $dP_t = \sigma dB_t + \alpha dt + \text{<impact>}$, where impact can be permanent and ~~the~~ temporary.

② Also, the price trajectory: $\tilde{P}_t = P_0 + \alpha t + \sigma W_t + v(X_t - X_0) + H(\theta_t)$

③ The trading cost: $Z = \int_0^T \tilde{P}_t \, dX_t = \int_0^T [P_0 + \alpha t + \sigma W_t + v(X_t - X_0) + H(\theta_t)] \theta_t dt.$

$= P_0 X + \frac{1}{2} v X^2 + \eta \int_0^T \theta_t^2 dt + \alpha \int_0^T t\theta_t dt + \sigma \int_0^T W(t) dX(t). =$ Non-random part + random part

To calculate the statistics.

$E(Z) = P_0 X + \frac{1}{2} v X^2 + \eta \int_0^T \theta_t^2 dt + \alpha \int_0^T t\theta_t dt \qquad V(Z) = \sigma^2 \int_0^T (X - X_t)^2 dt.$

④ the execution cost: $C = Z - P_0 X$.

⑤ we form the optimization problem: $\max_{\substack{X(0)=0 \\ X(T)=X}} E(Z) + \lambda V(Z)^{(*)}$, we want to reform the problem:

$\lambda\sigma^2 \int_0^T (X - (\xi(t) + \bar{x}))^2 dt.$

Let $\xi(t) = X(t) - \bar{x}$, $\quad (*) \Leftrightarrow \quad F(\xi) = \eta \int_0^T \xi'(t)^2 dt + \alpha \int_0^T t\xi'(t) dt + \lambda\sigma^2 \int_0^T (X - (\xi(t) + \bar{x}))^2 dt.$ is equivalent to have

To have $z(0) = z(T) = 0$. with perturbation $z(t)$. adding up to optimal $y(t)$. is equivalent to have

the linear part equaling 0 for any $z(t)$.

⑥ $F(\xi + z) - F(\xi) \sim \alpha \int_0^T t z'(t) dt + 2\eta \int_0^T \xi'(t) z'(t) dt + 2\lambda\sigma^2 \int_0^T T^2(t)(\xi(t) + \bar{x} - X) dt + O(z^2).$

Here, $\int_0^T \xi'(t) z'(t) dt = -\int_0^T z(t) \xi''(t) dt.$

$\int_0^T z'(t) t \, dt = -\int_0^T z(t) dt.$

$\therefore F(\xi + z) - F(\xi) \approx 2 \int_0^T [\lambda\sigma^2(\xi(t) + \bar{x} - X) - \eta \xi''(t) - \tfrac{1}{2}\alpha] z(t) dt + O(z^2).$

⑦ Keep the linear term 0: $\quad \lambda\sigma^2(\xi(t) + \bar{x} - X) - \eta\xi''(t) - \frac{1}{2}\alpha = 0.$

Just for intuition, we choose $\bar{x} = X + \frac{\alpha}{2\lambda\sigma^2}$ s.t. $\lambda\sigma^2(\bar{x} - X) - \frac{1}{2}\alpha = 0$; and $\xi''(t) = k^2 \xi(t)$

The solution for $\xi(t)$ that satisfies the boundary condition $\xi(0) = -\bar{x}$, $\xi(T) = X - \bar{x}$.

$$\xi(t) = -\frac{\alpha}{2\lambda\sigma^2} \frac{\sinh(k(T-t)) + \sinh(kt)}{\sinh(kT)} - X \frac{\sinh(k(T-t))}{\sinh(kT)}$$

$$x(t) = \xi(t) - \bar{x} = \frac{\alpha}{2\lambda\sigma^2}\left(1 - \frac{\sinh(k(T-t)) + \sinh(kt)}{\sinh(kT)}\right) + X\left(1 - \frac{\sinh(k(T-t))}{\sinh(kT)}\right)$$

⑧ When the static optimal portfolio $x^* = \frac{\alpha}{2\lambda\sigma^2}$ is larger than the target position X, we can likely have the trajectory temporarily exceeding the target position.

9 Some explanation:

A. $\bar{x} = X + \alpha^*$, Here, $\bar{x}$ is the optimal level the optimal trajectory trying to "reach".

And the $\xi(t)$ is the actual distance between the "optimal level" and current price level.

Also, $\bar{x}$ is ~~the part we~~ introduced to reduce the computational cost.

B. for $x(t) - \bar{X} > 0$. $\alpha^*(\sinh(kT) - \sinh(k(T-t)) - \sinh(kt)) > X\sinh(k(T-t))$

$$\Leftrightarrow \frac{\alpha^*}{X} > \frac{\sinh(k(T-t))}{(\sinh(kT) - \sinh(k(T-t)) - \sinh(kt))}$$

$\because \sinh(kT) = \sinh(k(T-t))\cosh(kt) + \cosh(k(T-t))\sinh(kt) \geq \sinh(k(T-t)) + \sinh(kt)$

For this equation, the RHS is positive. Therefore, if X is small enough, the optimal trajectory may exceed X and reverse the direction of trading.

C. If $\alpha < 0$, the optimal position is then ~~six~~ negative. Thus, the optimal trajectory may firstly short the security and then buy it back.