Quantitative Finance - Assingmnet I

Davi Barreira, Gabriel Monteiro, Pedro Dall

November 29, 2021

1 Algorithmic Trading: Incorporating Order Flux in Optimal Execution

We wish to liquidate a significant amount of an asset, such that our actions affect it's price. If we just try to sell everything at the current market price, our order will walk the book, and thus,we'll end up selling at a price possibly much lower. To avoid this problem, we wish to break our order in smaller orders, and "slowly" sell the asset, so that the Limit Order Book can be "replenished". The problem now is, if you take to long to sell the asset, you are subjected to the market's variation, and the asset's price might drop (which is probably what you expected, since you wanted to sell!). Hence, we are looking for an optimal strategy to sell the asset, i.e. how fast should we sell it? Also, we consider that our time-horizon is finite, i.e. we wish to liquidate all the asset's shares at most at time T.

1.1 Modeling the Problem

Let's define some of the variables:

- ν - The strategy to liquidate the asset;
- ν_t - The rate of which we sell the asset at time t.
- μ_t - The net order flow, i.e. the rate at which the market is buying/selling the asset;
- X_t^ν - The cash flow of the agent at time t using strategy ν;
- S_t^ν - The price of the asset at time t under strategy ν;
- Q_t^ν - Amount of "shares" of the asset the agent is holding at time t with strategy ν.

Our modelling assumptions are the following. First, we'll assume that the both our action and the market's affect the price of the assets linearly and equally, and that the price of the asset has a "natural" variability modeled as a Brownian motion with variance σ. Thus,the price of the asset is given by

$$dS_t^\nu = \sigma dW_t + b(\mu_t - \nu_t)dt,$$

where $b \in \mathbb{R}_+$, and W_t is the Brownian motion. The net order flow μ_t will be modelled as two independent Poisson processes, one for buying orders and another for selling orders with

$$d\mu_t = -\kappa\mu_t dt + \eta(dL_t^+ - dL_t^-)$$

where η is the impact of the increase/decrease of order flow, and κ is a rate of decay of the jumps. Our last simplifying assumption is that, when we sell at a rate ν_t, we always walk the book a bit, which is approximated by a linear equation $k\nu_t$ to model such impact, with $k > 0$. Thus, the cash process of the agent is given by:

$$dX_t^\nu = (S_t^\nu - k\nu_t)\nu_t dt.$$

Note that if we assumed that we always sell the midprice, then cash process would be simply $dX_t^\nu = S_t^\nu \nu_t dt$, i.e. the amount of money we get is the amount we trade times the price. The amount of shares we have can be computed as:

$$dQ_t^\nu = -\nu_t dt.$$

Lastly, we need to know what function we are going to try to optimize. This is a bit controversial. We might think that what we want to do is find $\nu \in \mathcal{A}$ (set of admissible strategies) that maximizes $E[X_T^\nu]$, i.e. maximize the expected value of the agent's cash at the final time T. But, if this was the case, the strategy is not penalized if it's too slow. Hence, instead of $E[X_T^\nu]$, we'll try to maximize

$$E\left[X_T^\nu + Q_T^\nu(S_T^\nu - \alpha Q_T^\nu) - \phi\int_t^T (Q_s^\nu)^2 ds\right].$$

Note that we added two extra terms. The first one was $Q_T^\nu(S_T^\nu - \alpha Q_T^\nu)$. This term penalizes the strategy if there is still some of the asset left by the end of time T. When this happens, the agent has to sell everthing at market price. But if we do that, the strategy is penalized by the amount of shares times a parameter $\alpha > 0$. Secondly, $\phi\int_t^T (Q_s^\nu)^2 ds$, which penalizes the strategy by how long it holds the asset. This is controlled by a parameter $\phi > 0$. The largest ϕ, the quicker the agent wants to get rid of the asset.

Ergodicity Economics. Even with some modifications, such optimization function still has a problem... Note that we are trying to maximize an expected value, but is this what we really want? Mathematicians and economists have noticed a longtime ago that, in practice, humans do not actually ”try” to maximize the expected value, but something else. Economists tried to fix this decoupling between human behavior and what seemed to be the correct mathematical function, and thus introduced what is called a utility function. This function is supposed to model our apparent risk-aversion behavior. Yet, this is still problematic, because the choice of such utility function seems arbitrary. Perhaps the best example of such oddity is revealed in the St. Petersburg Paradox. The paradox is the following. Consider a game in which a fair coin is tossed. If you get ”heads”, then you add two dollars to your pile. If you flip ”heads” again, you double the current amount in your pile, i.e. the two dollars becomes four, and so on. The game goes on until you flip ”tails”, in which the game ends and you receive the amount of money currently in the pile. The question is then, how much money would you pay to play this game? Here is where things get interesting.

If you try to calculate the expected value of this game, you'll find out that it's actually infinite. Hence, if the ”fair” price to play the game is the expected value, then the ”rational” thing to do is to play the game for any amount of money... You can already smell the b.s. in the air. We can guess that there is something off about this. It's not a surprise that this was coined as a paradox, since the ”logical” answer seems so ”unlogical”. This problem was posed in 1738 by Daniel Bernoulli, which proposed to solve it by introducing the ”utility function”. Yet, this seemed like cheating, since it's not clear how this notion could ”naturally” emerge from the original problem.

It was only in 2011 that the physicist Ole Peters proposed a new solution, which does not make use of any funky utility function, but instead, relies only in the problem itself. He realized that the St. Petersburg game is not ergodic, i.e.the time-average of the game is not equal to the ensamble-average (also known as ”expected” value). The only reason the game has infinite expected value is that a tiny fraction of possible games result in an absurd amount of money. But you, a single individual, would only be able to get such large amount of money if you played the game a zillion times... And if you actually added the amount of money you had to pay to play each time, you would not make a lot of money. Now we can more clearly see why the expected value of the cash process is not the ”correct” thing to maximize.

1.2 Dynamic Programming

Before solving our problem, let’s recap some of the results from Dynamic Programming.The Dynamic Programming Principle is formulated considering the following optimization problem:

$$H(t,x) = \sup_{\nu \in A} E\left[G(X_T^\nu) + \int_t^T F(s, X_s^\nu, \nu_s)ds\right].$$

If our process X is a diffusion given by $X_t^\nu = \mu(t, X_t^\nu, \nu_t)dt + \sigma(t, X_t^\nu, \nu_t)dW_t$, then theinifinitesimal generator of X_t^ν is

$$\mathcal{L}_t^\nu = \mu(t,x,\nu)\partial_x + \frac{1}{2}\sigma^2(t,x,\nu)\partial_{xx},$$

Now consider a process:

$$d\mu_t = -\kappa\mu_t dt + \eta_{1+N_{t-}} dN_t$$

where N_t is a homogeneous Poisson process with parameter λ and $\eta_i \sim \Xi$ are i.i.d random variables coming from a distribution with finite first moment (e.g. Exponential). For such process, the infinitesimal generator is given as the following (page 221 from book Algorithmic and High-Frequency Trading):

$$\mathcal{L}_t^\nu H(t,x,S,\mu,q) = -\kappa\mu\partial_\mu H + \lambda E_\Xi[H(t,x,S,\mu+\eta,q) - H(t,x,S,\mu,q)].$$

With the infinitesimal generators, we can postulate the **Hamilton-Jacobi-Bellman equation (HJB)**, which is:

$$\partial_t H(t,x) + \sup_{\nu \in \mathcal{A}}(\mathcal{L}_t^\nu H(t,x) + F(t,x,\nu)) = 0, \quad H(T,x) = G(x).$$

1.3 Analytical Solution

Remember, we want to find the optimal strategy $\nu^* \in \mathcal{A}$ such that

$$\begin{aligned}\nu^* &= \arg\sup_{\nu \in \mathcal{A}} E_{t,x,S,\mu,q}\left[X_t^\nu + Q_T^\nu(S_T^\nu - \alpha Q_T^\nu) - \phi\int_t^T (Q_s^\nu)^2 ds\right] \\ &= \arg\sup_{\nu \in \mathcal{A}} H^\nu(t,x,S,\mu,q).\end{aligned}$$

Our Stochastic Control Problem is:

$$\begin{cases} H(t,x) := \sup_{\nu\in\mathcal{A}} H^\nu(t,x,S,\mu,q), \\ X_t^\nu, \text{ solves } dX_t^\nu = (S_t^\nu - k\nu_t)\nu_t dt, \\ X_0^\nu = 0, \\ dS_t^\nu = b(\mu_t - \nu_t)dt + \sigma dW_t, \\ S_0^\nu = S_0, d\mu_t = -\kappa\mu_t dt + \eta(dL_t^+ - dL_t^-), \\ dQ_t^\nu = -\nu_t dt, \\ Q_0^\nu = Q_0. \end{cases}$$

Let's use Dynamic Programming to solve our modelling problem. The **HJB** equation that we presented before werederived for<!– The Dynamic Programming Principle is formulated for the problem optimization problem formulated with the following expression: –>

$$H(t,x) = \sup_{\nu\in A} E\left[G(X_T^\nu) + \int_t^T F(s, X_s^\nu, \nu_s)ds\right]$$

In our case,

$$G(X_T^\nu) = X_t^\nu + Q_T^\nu(S_T^\nu - \alpha Q_T^\nu), \quad F(s, X_s^\nu, \nu_s) = -\phi(Q_s^\nu)^2.$$

To obtain the HJB equation, we have to calculate $\mathcal{L}_t^\nu$ for each of variable, i.e. x, S, μ, q.Let's use the formula for the diffusion problem which we presented above in the section **Dynamic Programming**:

$$\begin{aligned} \mathcal{L}_{t,S}^\nu H &= b(\mu - \nu)\partial_S H + \frac{1}{2}\sigma^2\partial_{SS}H, \\ \mathcal{L}_{t,x}^\nu H &= (S - k\nu)\nu\partial_x H \\ \mathcal{L}_{t,q}^\nu H &= -\nu\partial_q H. \end{aligned}$$

For μ, use generator obtained from the mean-reverting process with the Poisson jumps. In our case, we actuallyhave two independent Poisson processes (a positive and a negative):

$$\begin{aligned} \mathcal{L}_{t,\mu}^\nu H = -\kappa\mu\partial_\mu H &+ \lambda E[H(t,x,S,\mu+\eta,q) - H(t,x,S,\mu,q)]] \\ &+ \lambda E[H(t,x,S,\mu-\eta,q) - H(t,x,S,\mu,q)] \end{aligned}$$

The *HJB* equation for our problem becomes:

$$\begin{aligned} 0 &= \partial_t H + \sup_{\nu \in \mathcal{A}} \{ \mathcal{L}^{\nu}_{t,S} + \mathcal{L}^{\nu}_{t,x} + \mathcal{L}^{\nu}_{t,\mu} + \mathcal{L}^{\nu}_{t,q} + (-\phi q^2) \} \\ &= \partial_t H + \frac{1}{2}\sigma^2 \partial_{SS} H + \mathcal{L}^{\nu}_{t,\mu} H + (-\phi q^2) \\ &\qquad + \sup_{\nu \in \mathcal{A}} \{ b(\mu - \nu)\partial_S H + (S - k\nu)\nu \partial_x H + -\nu \partial_q H \} \end{aligned}$$

To solve this, let's do an educated guess and assume that

$$H(t, x, S, \mu, q) = x + qS + h(t, \mu, q).$$

Note that this comes from our boundary condition, which is

$$H(T, x, S, \mu, q) = x + qS - \alpha q^2.$$

Hence, we are assuming that the first two terms are equal to the values in the boundary, and we must find a function $h(t, \mu, q)$, such that $h(T, \mu, q) = -\alpha q^2$. Substituting this H in the HJB equation above, we get:

$$\partial_t h + \mathcal{L}^{\nu}_{t,\mu} h + b\mu q - \pi q^2 + \sup_{\nu \in \mathcal{A}} \{ -k\nu^2 - (bq + \partial_q h)\nu \} = 0.$$

To calculate $\sup_{\nu \in \mathcal{A}} \{ -k\nu^2 - (bq + \partial_q h)\nu \}$, we take the derivative with respect to ν and make it equal to zero.

$$\frac{\partial}{\partial \nu} - k\nu^2 - (bq + \partial_q h)\nu = -2k\nu - bq - \partial_q h = 0 \implies \nu^* = -\frac{1}{2k}(bq + \partial_q h).$$

Substituting this value of ν in the PDE, we get:

$$\partial_t h + \mathcal{L}^{\nu}_{t,\mu} h + b\mu q - \phi q^2 + \frac{1}{4k}(bq + \partial_q h)^2 = 0.$$

We need to solve this equation above for h. Again we do an educated guess, assuming that:

$$h(t, \mu, q) = h_0(t, \mu) + q h_1(t, \mu) + q^2 h_2(t, \mu).$$

Since our boundary condition is $h(T, \mu, q) = -\alpha q^2$, then $h_0(T, \mu) = h_1(T, \mu) = 0$ and $h_2(T, \mu) = -\alpha$. Using this on the PDE, we get

$$\begin{aligned} &(\partial_t h_0 + \mathcal{L}^{\nu}_{t,\mu} h_0 + \frac{1}{4k} h_1^2) + \\ &q(\partial_t h_1 + \mathcal{L}^{\nu}_{t,\mu} h_1 + \frac{1}{2k} h_1 (b + 2h_2) + \\ &q^2(\partial_t h_2 + \mathcal{L}^{\nu}_{t,\mu} h_2 - \phi + \frac{1}{4k}(b + 2h_2)^2 = 0. \end{aligned}$$

The terms that are function of q^2 only depend on h_2, so we can solve for h_2 independently. Also, note that since $h_2(T,\mu) = -\alpha$ and that no there is no other term with μ, we can conclude that h_2 is independent for μ. Hence,

$$\begin{aligned}\mathcal{L}^{\nu}_{t,\mu} h(T,\mu) &= -\kappa\mu\partial_\mu h(T,\mu) + \lambda E[h(T,\mu+\eta) - h(T,\mu)] \\ &\quad + \lambda E[h(T,\mu-\eta) - h(T,\mu)] \\ &= 0.\end{aligned}$$

Therefore, the equation in terms of h_2 reduces to:

$$0 = \partial_t h_2(t) - \phi + \frac{1}{4k}(b + 2h_2(t))^2,$$

subjected to $h_2(T) = -\alpha$. This problem is an ODE of Riccati type, and has an exact solution of

$$h_2(t) = \chi(t) - \frac{1}{2}b,$$

such that

$$\chi(t) = \sqrt{k\phi}\frac{1 + \zeta e^{2\gamma(T-t)}}{1 - \zeta e^{2\gamma(T-t)}}, \quad \gamma = \sqrt{\frac{\phi}{k}}, \quad \zeta = \frac{\alpha - b/2 + \sqrt{k\phi}}{\alpha - b/2 - \sqrt{k\phi}}.$$

Now that we know h_2, we can solve the part of the PDE that is multiplied by q, i.e.

$$\partial_t h_1 + \mathcal{L}^{\nu}_{t,\mu} h_1 + b\mu \frac{1}{2k} h_1 (b + 2 \underbrace{h_2}_{\chi(t) - \frac{1}{2}b}) = 0.$$

Suppose that $h_1(t,\mu) = \ell_0(t) + \mu\ell_1(t)$, thus,

$$\mathcal{L}^{\nu}_{t,\mu} h_1 = -\kappa\mu\ell_1 + E[\lambda(\eta,\ell) + \lambda(-\eta\ell_1)] = -\kappa\mu\ell_1,$$

subjected to $\ell_0(T) = \ell_1(T) = 0$. We therefore obtain that

$$\partial_t \ell_0 + \frac{1}{k}\chi(t)\ell_0 + \mu\left[\partial_t \ell_1 + \ell_1\left(\frac{1}{k}\chi(t) - \kappa\right) + b\right]$$

1.4 Simulation

$$\gamma = \sqrt{\frac{\phi}{k}}$$

$$\zeta = \frac{\alpha - b/2 + \sqrt{k\phi}}{\alpha - b/2 - \sqrt{k\phi}}$$

$$\chi(t) = \sqrt{k\phi}\frac{1 + \zeta e^{2\gamma(T-t)}}{1 - \zeta e^{2\gamma(T-t)}}$$

$$\nu_t^* = -\frac{1}{k}\chi(t)Q_t^{\nu^*} - \frac{b}{2k}\bar{\ell}_1(t)\mu_t$$

where ℓ_1^t can be expressed as

$$\ell_1(t) = b\int_t^T e^{-\kappa(s-t)}e^{\frac{1}{k}\int_t^s \chi(u)du}ds$$

and simplifies to

$$\ell_1(t) = b\,\ell(T-t) \geq 0$$

where

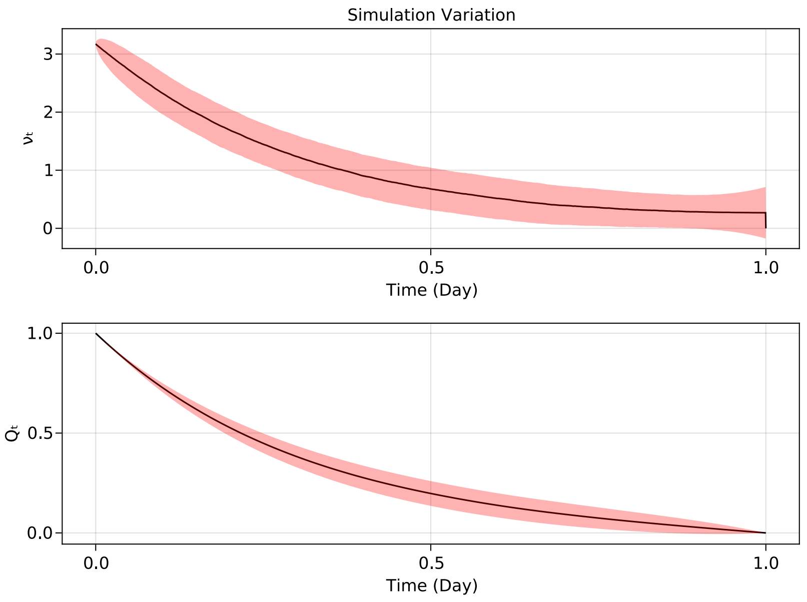
$$\ell(\tau) = \frac{1}{\zeta e^{\gamma\tau} - e^{-\gamma\tau}}\left\{e^{\gamma\tau}\frac{1 - e^{-(\kappa+\gamma)\tau}}{\kappa+\gamma}\zeta - e^{-\gamma\tau}\frac{1 - e^{-(\kappa-\gamma)\tau}}{\kappa-\gamma}\right\}$$

1.5 Computing the constants

1.6 Initializing Variables

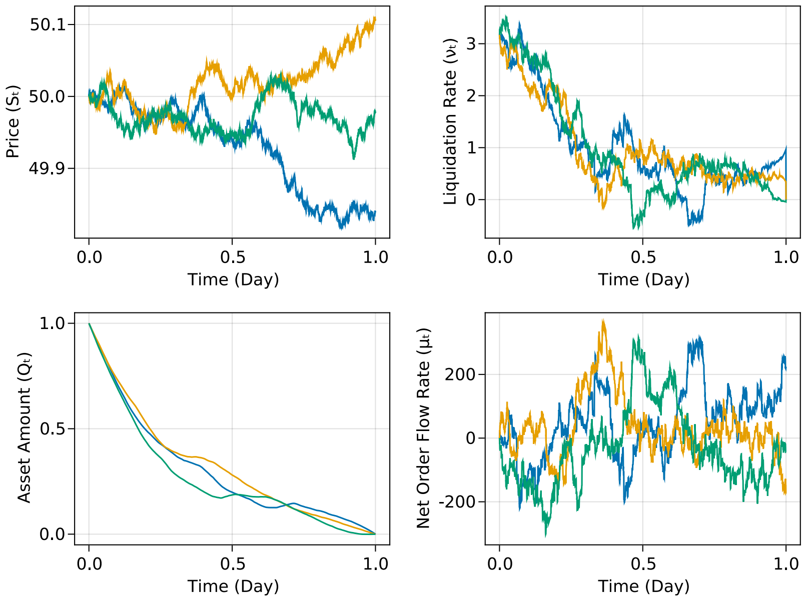
```julia
function bandplot(X,t)
    Xₘ = mean(X,dims=1)'[:]
    Xₛ = std(X,dims=1)'[:];
    plot = lines(t, Xₘ, color=:black)
    band!(t, Xₘ + Xₛ ,Xₘ - Xₛ,  color = (:red, 0.3))
    return plot
end
```

```julia
f = Figure()

ax1 = Axis(f[1,1],xlabel="Time (Day)", ylabel="νₜ", title=
    "Simulation Variation")
νₘ = mean(ν,dims=1)'[:]
νₛ = std(ν,dims=1)'[:];
lines!(ax1, t, νₘ, color=:black)
band!(ax1, t, νₘ + νₛ ,νₘ - νₛ,  color = (:red, 0.3))
ax2 = Axis(f[2,1],xlabel="Time (Day)", ylabel="Qₜ")
Qₘ = mean(Q,dims=1)'[:]
Qₛ = std(Q,dims=1)'[:];
lines!(ax2, t, Qₘ, color=:black)
band!(ax2, t, Qₘ + Qₛ ,Qₘ - Qₛ,  color = (:red, 0.3))
f
```



```julia
f = Figure()

ax1 = Axis(f[1,1],xlabel="Time (Day)", ylabel="Price (Sₜ)"
    )
```

```julia
lines!(ax1, t, S[1,:])
lines!(ax1, t, S[2,:])
lines!(ax1, t, S[3,:])

ax2 = Axis(f[2,1],xlabel="Time (Day)", ylabel="Asset
    Amount (Qₜ)")
lines!(ax2, t, Q[1,:])
lines!(ax2, t, Q[2,:])
lines!(ax2, t, Q[3,:])

ax3 = Axis(f[1,2],xlabel="Time (Day)", ylabel="Liquidation
     Rate (νₜ)")
lines!(ax3, t, ν[1,:])
lines!(ax3, t, ν[2,:])
lines!(ax3, t, ν[3,:])

ax4 = Axis(f[2,2],xlabel="Time (Day)", ylabel="Net Order
    Flow Rate (μₜ)")
lines!(ax4, t, μ[1,:])
lines!(ax4, t, μ[2,:])
lines!(ax4, t, μ[3,:])
f
```


50.1
50.0
49.9
Price (S_t)
0.0
0.5
1.0
Time (Day)
3
2
1
0
Liquidation Rate (ν_t)
Time (Day)
1.0
0.5
0.0
Asset Amount (Q_t)
Time (Day)
200
0
-200
Net Order Flow Rate (μ_t)
Time (Day)