A Dynamic Approach for Optimal Order Execution based on VWAP Benchmark using Stochastic Control

Yuqi Jing

September 28, 2022

Abstract

We study the distribution of intraday market volume under the benchmark of VWAP. Our objective is to increase the tracking fidelity of VWAP by improving the standard method used in brokerage industry, in particular by incorporating the information coming from the market during the trading in the best possible way. The assessing mechanism is defined as minimising the RMSE of the slippage, with quadratic transaction costs. We evaluate our method with extensive simulation of order execution on real Nasdaq-100 market data. Our proposed solution, using a new model for market volumes, reduces by 10% the RMSE of VWAP tracking compared to the original solution and reduce transaction costs by 21%.

1 Introduction

Trading is central to investment process. Among various trading techniques to reduce transaction cost, the VWAP benchmarking, is the most fundamental and popular empirical methodologies applied in the quantitative funds [Mad02]. Using this benchmark makes the problem more appealing from a stochastic control standpoint and motivates the development of numerous statistical models for the market dynamics [Mad02]. The optimal order execution based on VWAP has been studied originally with a fixed optimisation perspective [Kon02], [OW05], i.e. the volume model is fixed at the beginning of every trading schedule.

However, intuitively, this is not optimal, since the volume model does not incorporate the new information coming from the market during the trading schedule [HK11]. A recent paper [GR13] studies the problem from a stochastic control standpoint. Our solution draw inspiration from the work with a key difference: we study the optimisation problem under the frame of a recent study in optimal control [SBZ10] to avoid the curse of dimensionality, hence greatly improve the computational efficiency of the work in [GR13]. We also provide extensive simulations results to validate our work.

We formulate the problem and all the relevant variables in Sec 2. In Sec 3 we define our improved dynamic solution. In Sec 4 we present the simulation results we conducted, on Nasdaq-100 index, with our dynamic solution. We conclude in Sec 5 while also raising question regarding the performance of the solution in small-cap stocks for further research and validation.

2 Problem formulation

2.1 Definition

We restrict our analysis to "buy" order in context. We first consider the *slippage* which is defined as the amount by which the order execution price misses the benchmark (in this case our benchmark is p_{vwap}).The trading industry ususally defines the slippage as

$$\sum_{t=1}^{T} u_t \hat{p}_t - C p_{vwap}$$

where u_t is the traded volume of the order in *time interval* t, $\hat{p}_t$ is the *effective* price defined so that the whole cost of the trade at interval t is $u_t \hat{p}_t$ ($\hat{p}_t$ represents *transaction costs* and *market impact*). C is target volume. p_{vwap} is benchmark price which is computed as $\sum_{t=1}^{T} m_t p_t / V$.

For variables in

$$\frac{\sum_{t=1}^{T} m_t p_t}{V}$$

p_t is the average price in time interval t (there are still disputes regarding the derivation of p_t which is preferably interpreted as interval vwap). m_t is the total traded volume in time interval t. V is the total traded volume in that day.

We instead define slippage as

$$S = \frac{\sum_{t=1}^{T} u_t \hat{p}_t - C p_{vwap}}{C p_{vwap}}, \tag{1}$$

normalized by the value of the order.

We know in reality the broker trades the volume u_t using an algorithm which mixes market and limit orders. Let $s_t \in \mathbb{R}_{++}$ be the average fractional (as ratio of the stock price) bid-ask spread in period t. Therefore, the cost per share of a buy market order is on average $p_t(1 + s_t/2)$ while for a limit buy order it is on average $p_t(1 - s_t/2)$.

Some assumptions for the problem formulation are:

- We assume that the fraction of market orders over the traded volume is proportional to the *participation rate*, defined as u_t/m_t. So

  $$\frac{u_{MO}}{u_t} = \frac{\alpha}{2} \frac{u_t}{m_t}.$$

  This is a reasonable assumption since the stocks in Nasdaq-100 we examine is highly liquid hence has extremely small participation rate.

- Here we choose α as 90 following the assumption that the fractional spread s_t is constant in time and equal to 2 basis points (0.0002), i.e. $s_1 = \cdots = s_T = 2b.p$ which is introduced for efficiency purposes in calculation later.

- Trading one day's volume costs approximately one day's volatility. We estimate empirically over the first 20 days of Nasdaq-100 of our sample the open-to-close volatility, equal to 90 points.

The whole cost of the trade is

$$u_t \hat{p_t} = p_t \left(u_{LO}(1 - \frac{s_t}{2}) + u_{MO}(1 + \frac{s_t}{2}) \right),$$

which implies

$$\hat{p_t} = p_t \left(1 - \frac{s_t}{2} + \alpha \frac{s_t}{2} \frac{u_t}{m_t} \right).$$

We thus have a simple model for the effective price $\hat{p_t}$, linear in u_t. This gives rise to a *quadratic* transaction costs, a reasonable approximation consistent with the trading industry ([BFL09]). By substituting the $\hat{p_t}$ defined above into slippage we get the expression for S (where we had two approximations, both first order), so that our objective function is to minimize the mean-variance of S

$$\mathbf{E}S + \lambda \mathbf{var}(S)$$

for a given risk-aversion parameter $\lambda \geq 0$. These expectation and variance operators apply to all sources of randomness in the system, *i.e.*, the market volumes m and market prices p, which are independent under our model. The expected value of the slippage is

$$\mathop{\mathbf{E}}_{m,p} S = \mathop{\mathbf{E}}_{m} \left[\sum_{t=1}^{T} \frac{s_t}{2} \left(\alpha \frac{u_t^2}{C m_t} - \frac{u_t}{C} \right) \right]. \tag{2}$$

The variance of the slippage is

$$\mathop{\mathbf{E}}_{m} \mathop{\mathbf{var}}_{p} = \mathop{\mathbf{E}}_{m} \left[\sum_{t=1}^{T-1} \sigma_{t+1}^2 \left(\frac{\sum_{\tau=1}^{t} m_t}{V} - \frac{\sum_{\tau=1}^{t} u_t}{C} \right)^2 \right]. \tag{3}$$

We thus get

$$\mathbf{E}_{m,p} S + \lambda \mathbf{var}_{m,p}(S) = \sum_{t=1}^{T} \mathbf{E}_m \left[\frac{s_t}{2} \left(\alpha \frac{u_t^2}{C m_t} - \frac{u_t}{C} \right) + \lambda \sigma_t^2 \left(\frac{\sum_{\tau=1}^{t-1} m_t}{V} - \frac{\sum_{\tau=1}^{t-1} u_t}{C} \right)^2 \right]. \tag{4}$$

2.2 Solution for Optimisation

We start to solve the optimization problem with objective function and two constraints

$$\begin{aligned} \text{minimize}_u \quad & \mathbf{E}_{m,p} S + \lambda \mathbf{var}_{m,p}(S) \\ \text{s.t.} \quad & \sum_{t=1}^{T} u_t = C \\ & u_t \geq 0, \ t = 1, \ldots, T \end{aligned}$$

We write the problem in the equivalent form

$$\begin{array}{ll}\text{minimize}_u & \displaystyle\sum_{t=1}^{T}\mathbf{E}_m\left[\frac{s_t}{2C}\left(\alpha {u_t}^2 k_t - u_t\right) + \lambda\sigma_t^2\left(\left(\frac{\sum_{\tau=1}^{t-1}u_t}{C}\right)^2 - 2M_t\frac{\sum_{\tau=1}^{t-1}u_t}{C}\right)\right] \\ \text{s.t.} & \displaystyle\sum_{t=1}^{T}u_t = C \\ & u_t \geq 0,\ t = 1,\ldots,T\end{array}$$

where M_t and k_t are the constants,

$$M_t = \mathop{\mathbf{E}}_m\left[\sum_{\tau=1}^{t-1}m_t\right],\quad k_t = \mathop{\mathbf{E}}_m\left[\frac{1}{m_t}\right]$$

for $t = 1,\ldots,T$. In this form, the problem is a standard quadratic program which can be solved by open-source solvers using a symbolic convex optimization suite.

We consider the special case of constant spread, $s_1 = \cdots = s_T$, which leads to a great simplification of the solution. The convex problem has the form

$$\begin{array}{ll}\text{minimize}_u & \sum_{t=1}^{T}\frac{s_t}{2C}\left(\alpha u_t^2\kappa_t - u_t\right) + \lambda\left(\sum_{t=1}^{T}\sigma_t^2\left(U_t^2 - 2M_tU_t/C\right)\right) \equiv \phi(u) + \lambda\psi(u) \\ \text{s.t.} & u \in \mathcal{C}\end{array}$$

where $U_t = \sum_{\tau=1}^{t-1}u_t/C$ for each $t = 1,\ldots,T$. We separate the problem into two subproblems considering each of the two terms of the objective. The first one is

$$\begin{array}{ll}\text{minimize} & \phi(u) \\ \text{s.t.} & u \in \mathcal{C}\end{array}$$

which is equivalent to

$$\begin{array}{ll}\text{minimize} & \displaystyle\sum_{t=1}^{T}u_t^2k_t \\ \text{s.t.} & u \in \mathcal{C}\end{array}$$

The optimal solution by Lagrange duality is

$$u_t^\star \simeq C\frac{\mathbf{E}_m[m_t]}{\sum_{t=1}^{T}\mathbf{E}_m[m_t]} \simeq C\mathop{\mathbf{E}}_m\left[\frac{m_t}{V}\right],\quad t = 1,\ldots,T.$$

The second problem is

$$\begin{array}{ll}\text{minimize}_u & \psi(u) \equiv \sum_{t=1}^{T}\sigma_t^2\left(U_t^2 - 2M_tU_t/C\right) \\ & u \in \mathcal{C}\end{array}$$

we choose the U_t, $U_t = \sum_{r=1}^{t-1}u_t/C$ such that $\sigma^2(U_t - M_T) = 0$ so $U_t = M_t$. The values of $u_1, u_2, \ldots, u_{t-1}$ are thus fixed and these u_t values are feasible. It follows that this U_t is an optimal solution, it has values

$$u_t^* = C\,\mathbf{E}_m\left[\frac{m_t}{V}\right].$$

We now consider the original problem. The objective function is a convex combination of two convex problems above and all three have the same constraints set. Since the two subproblems share an optimal solution u^*, it follows that u^* is also an optimal solution for the combined problem. Thus, an optimal solution of objective function in the case of constant spread is

$$u_t^* = C\,\mathbf{E}_m\left[\frac{m_t}{V}\right] \quad t\,=\,1,\ldots,T. \tag{5}$$

This is standard in the brokerage industry. In practice we could derive more sophisticated solutions (estimating $\mathbf{E}_m[1/m_t]$ would require a more sophisticated model of market volumes than $\mathbf{E}[m_t/V]$). However, with respect to minimisation of the variance of S, this solution is indeed optimal.

3 The Dynamic Approach

We develop a *dynamic* approach that incorporates all the information coming from the market during the schedule, based on study in [GR13]. We develop a new intraday volume model, summarised in Sec 3.1. We work in the framework of Dynamic Programming (DP) [Ber95] for optimisation of the system, described in Sec 3.2. In particular we strive to avoid the curse of dimensionality (i.e. tackle the computational difficulty in the original method in [GR13]) by using the approximate procedure of [SBZ10], summarised in Sec 3.3. In particular we fit the problem in the special case of linear dynamics and quadratic costs, described in Sec 3.4.

3.1 Intraday Volume Model

We propose a new intraday volume model with the distribution of a multivariate log-normal.

$$f_m(m_1, m_2, \ldots, m_N) \sim \ln\mathcal{N}(\mu\ + \mathbf{1}b, \mathbf{\Sigma})$$

where $b \in \mathbf{R}$ is a constant that depends on the security k (each security has a different typical daily volume), $\mu \in \mathbf{R}^T$ is an average "volume profile" (normalised so that $\mathbf{1}^T\mu = 0$). The new model takes into account the conditional dependence between disturbances of volumes in N time intervals, for which N is equal to 390, so that each interval is one minute long.

Estimation of $\sum$ We use model parameters calibrated on historical data exclusively. In other words the performance of our model are estimated *out-of-sample*. To avoid overfitting (because of the high dimension N), We utilize an approximation approach in which we look for a matrix of the form

$$\mathbf{\Sigma} = ff^T + S,$$

where $f \in \mathbf{R}^N$ and S is sparse.

We first build the empirical covariance matrix $\hat{\Sigma}$ where we use sigular value decomposition of X

$$X = U \cdot \mathbf{diag}(s_1, s_2, \ldots, s_T) \cdot V^T,$$

and choose the best singular value based on data. We thus approximate the ff^T with the best rank-1 approximation of empirical covariance matrix. We assume that S is a *banded* matrix of bandwidth $b > 0$. We then find S by copying the diagonal elements of the empirical covariance matrix.
We thus have built a matrix of the form $\Sigma = ff^T + S$. This does not guarantee Σ to be positive definite. However in our empirical tests we always got positive definite Σ for any b.

3.2 Dynamic Programming

We present here the formalism of dynamic programming for the optimisation of our intraday volume model, following [Ber95]. Suppose we have a state variable $x_t \in \mathcal{X}$ defined for $t = 1, \ldots, T+1$ with x_1 known. Our decision variables are $u_t \in \mathcal{U}$ for $t = 1, \ldots, T+1$ and each u_t is chosen as a function of the current state, $u_t = \mu_t(x_t)$. The randomness of the system is modeled by a series of IID random variables $w_t \in \mathcal{W}$ (disturbances). The dynamics is described by a series of functions

$$x_{t+1} = f_t(x_t, u_t, w_t),$$

at every stage we incur the cost

$$g_t(x_t, u_t, w_t),$$

and at the end of the decision process we have a final cost

$$g_{T+1}(x_{T+1}),$$

Our objective is to minimise

$$\mathcal{J} = \mathbf{E}\left[\sum_{t=1}^{T} g_t(x_t, u_t, w_t) + g_{T+1}(x_{T+1})\right].$$

We solve the problem by *backward induction*, defining the *cost-to-go* function v_T at each time step t

$$v_t(x) = \min_u \mathbf{E}\Big[g_t(x_t, u_t, w_t) + v_{t+1}(f_t(x_t, u_t, w_t))\Big], \quad t = 1, \ldots, T.$$

The recursion is known as *Bellman* equation. The final condition is fixed by

$$v_{T+1}(\,\cdot\,) = g_{T+1}(\,\cdot\,).$$

It follows that the optimal action at time t is given by the solution

$$u_t = \arg\min_u \mathbf{E}\Big[g_t(x_t, u_t, w_t) + v_{t+1}(f_t(x_t, u_t, w_t))\Big], \quad t = 1, \ldots, T. \tag{6}$$

In general, these equations require an enormous amount of computation (exponential in the dimension of the state space), action space, and number of time steps (*curse of dimensionality*). In addition, we intend to include in the conditionally dependent disturbances observed up to time t, which will *augment* the state x_t and causes further computational complexity of the solution. However, some *approximate dynamic programming* techniques can be used to solve the augmented problem [Pow07].

3.3 Under the Framework of SHDP

We take the approximate approach developed in [SBZ10], called *shrinking horizon dynamic programming* (SHDP), to deal with the computational problem we face in Dynamic Programming, while updating the distribution of the volume model via conditioning (by incorporating new information). It performs reasonably well in practice and leads to a tractable solution. We now summarise the approach. Assume we know the density of the future disturbances $w_t, \ldots, w_T$ conditioned on the observed ones

$$f_{w|t}(\cdot) : \mathcal{W} \times \cdots \times \mathcal{W} \to [0, 1],$$

which can be derived from the distribution of intrady volume model. We derive the marginal density of each future disturbance, by integrating over all others,

$$\hat{f}_{w_t|t}(w_t), \ldots, \hat{f}_{w_T|t}(w_T).$$

We then compute the cost-to-go functions with backwards induction using the Bellman equations, where the expectations over each disturbance w_r are taken on the conditional marginal density $\hat{f}_{w_t|t}(w_t)$. The equations for the cost-to-go function become

$$v_{\tau|t}(x) = \min_u \mathop{\mathbf{E}}_{\hat{f}_{w_\tau|t}} \Big[g_\tau(x, u, w) + v_{\tau+1|t}(f_\tau(x, u, w)) \Big],$$

for all times $\tau = t, \ldots, T$, with the usual final condition. Similarly, the equations for the optimal action become

$$u_{\tau|t}(x) = \arg\min_u \mathop{\mathbf{E}}_{\hat{f}_{w_\tau|t}} \Big[g_\tau(x, u, w) + v_{\tau+1|t}(f_\tau(x, u, w)) \Big], \tag{7}$$

for all time $\tau = t, \ldots, T$. We only use the solution u_t at time t. When we proceed to the next time step $t + 1$ we rebuild the whole sequence of cost-to-go functions $v_{t+1|t+1}(x), \ldots, v_{T|t+1}(x)$ using the updated marginal conditional densities and then solve the optimal action to get u_{t+1}. With this framework we can solve the VWAP problem we developed in Sec 2.

3.4 Linear-quadratic stochastic control

We continue with optimal action equation (1) in Dynamic Programming under the framework of SHDP. We adapt the dynamics functions f_t to be stochastic affine and the cost functions to be stochastic quadratic, so that the problem in Sec 3.1 has an analytic solution [BLR12]. It is called *Linear Quadratic Stochastic Control* (LQSC). We define the state space $\mathcal{X} = \mathbf{R^n}$, the action space $\mathcal{U} = \mathbf{R}^m$ for some $n, m > 0$. For $t = 1, \ldots, T$ the system dynamics is described by

$$x_{t+1} = f_t(x_t, u_t, w_t) = A_t(w_t)x_t + B_t(w_t)u_t + c_t(w_t)$$

with matrix function $A_t(\,\cdot\,) : \mathcal{W} \to \mathbf{R}^{n\times n}$, $B_t(\,\cdot\,) : \mathcal{W} \to \mathbf{R}^{n\times m}$, and $c_t(\,\cdot\,) : \mathcal{W} \to \mathbf{R}^n$. The cost functions are

$$g_t(x_t, u_t, w_t) = x_t^T Q_t(w_t)x_t + q_t(w_t)^T x_t + u_t^T R_t(w_t)u_t + r_t(w_t)^T u_t$$

with matrix function $Q_t(\,\cdot\,) : \mathcal{W} \to \mathbf{R}^{n\times n}$, $q_t(\,\cdot\,) : \mathcal{W} \to \mathbf{R}^n$, $R_t(\,\cdot\,) : \mathcal{W} \to \mathbf{R}^{m\times m}$, and $r_t(\,\cdot\,) : \mathcal{W} \to \mathbf{R}^m$. The final cost is a quadratic function of the final state

$$g_{T+1}(x_{T+1}) = x_{T+1}^T Q_{T+1} x_{T+1} + q_{T+1}^T x_{T+1}.$$

By combining the formalism with the machinery described in Sec 3.3, we provide an approximate solution of the problem defined in Sec 2 under the frame of dynamic programming. Consider a fixed time $t = 1, \ldots, T-1$. We note that we only observe the sequence of market volumes $m_1, \ldots, m_{t-1}$. Let $f_m(m_1, \ldots, m_T)$ be the joint distribution of the market volumes. Our market volume model also provides the conditional density $f_{m|t}(m_t, \ldots, m_T)$ of $m_1, \ldots, m_{t-1}$. We thus estimate the our market volume where the conditional distribution of V $f_{V|t}(\,\cdot\,)$ given $m_1, \ldots, m_{t-1}$. Let the marginal densities be

$$\hat{f}_{m_t|t}(m_t), \ldots, \hat{f}_{m_T|t}(m_T).$$

The marginal conditinal densities of the disturbances are thus

$$\hat{f}_{w_\tau|t}(\,\cdot\,) = \hat{f}_{m_\tau|t}(\,\cdot\,) \times f_{V|t}(\,\cdot\,)$$

for $\tau = t, \ldots, T$. We use these to apply the machinery of Sec 3.4, solve the Bellman equation and obtain the optimal SHDP policy at time t. The optimal policy $\mu_t(x_t)$ in linear-quadratic problem is therefore

$$\mu_t^* = \mu_{\tau|t}(x_t) = K_{\tau|t} x_t + l_{\tau|t}, \;\; t = 0, \ldots, T-1, \tag{8}$$

where $K_{\tau|t} \in \mathbf{R}^{m\times n}$ and $l_{\tau|t} \in \mathbf{R}^m$ depend on the problem parameters. In addition, the cost-to-go function is a quadratic function of the state

$$v_{\tau|t}(x_t) = x_t^T D_{\tau|t} x_t + d_{\tau|t}^T x_t + b_{\tau|t}, \tag{9}$$

where $D_{\tau|t} \in \mathbf{R}^{m\times n}$ and $b_{\tau|t} \in \mathbf{R}^m$. We derive these results solving the Bellman equations by backward induction. We then move to the next time step and repeat the whole process.

4 Simulation

The biggest contribution of this study is the extensive empirical results we obtain by studying the performance of the *dynamic solution* versus the the original method. We use historical Nasdaq-100 index trading data as our dataset. We organise our simulations according to a *rolling test* or *moving average* procedure: for every used to simulate order execution we estimate the various parameters on data from a window covering the preceding $W > 0$ days. Here we use $W = 20$. We thus simulate execution on each day $i = W+1, \ldots, N$ using data from the days $i - W, \ldots, i-1$ for historical estimation. We describe in Sec 4.1 the dataset and how we process it. In Sec 4.2 we summarise the formalism of standard solution. The dynamic solution requires model fitting by estimating the model parameters, explained in Sec 4.3. In Sec 4.4, we show our aggregate results and the improvement of DS over standard solution. Finally, in Sec 4.5 we analyse the aggregate results from the perspective of time scale effect.

4.1 Data

For previous studies in the relevant literature, various combinations of dataset have been studied, for example 30 different stocks in NYSE stock market[GR13], [BV09]. We pioneeringly used stock index as the simulating data. We chose Nasdaq-100 as our simulating data. It consists of the companies that have largest capitalisation and are most liquid in the US (also global) market. In addition, the index spans 101 largest companies from all categories of industries, which is a useful tool for assessing the performance of the dynamic solution.

Specifically, we consider the $N = 60$ market days corresponding to the second quarter of 2022, from April 1 to June 28. We use raw minutely Trading Data (TAQ) from Eikon Dataset to obtain daily series of market volumes $m_t \in \mathbf{Z}_+$ and average period price $p \in \mathbf{R}_+$, for $t = 1, \ldots, T$ where $T = 390$, with each interval one minute long. We focus exclusively on the continuous trading activity without considering market opening and closing nor any OTC trade.

4.2 The Standard Solution

The standard solution [Kon02] relies on the feature we derived in Sec 2 Equation 5.

$$u_t^* = C\,\mathbf{E}_m\left[\frac{m_t}{V}\right],$$

by calculating the expectation. We use the sample average

$$\mathbf{E}\left[\frac{m_t}{V}\right] \simeq \frac{\sum_{j=i-W}^{i-1} m_t^j / V^j}{W}$$

for every $t = 1, \ldots, T$. In Figure 1 we show an example of the estimation u_t versus real volume ratio m_t/V in $T = 390$ in one of the simulation trading days.

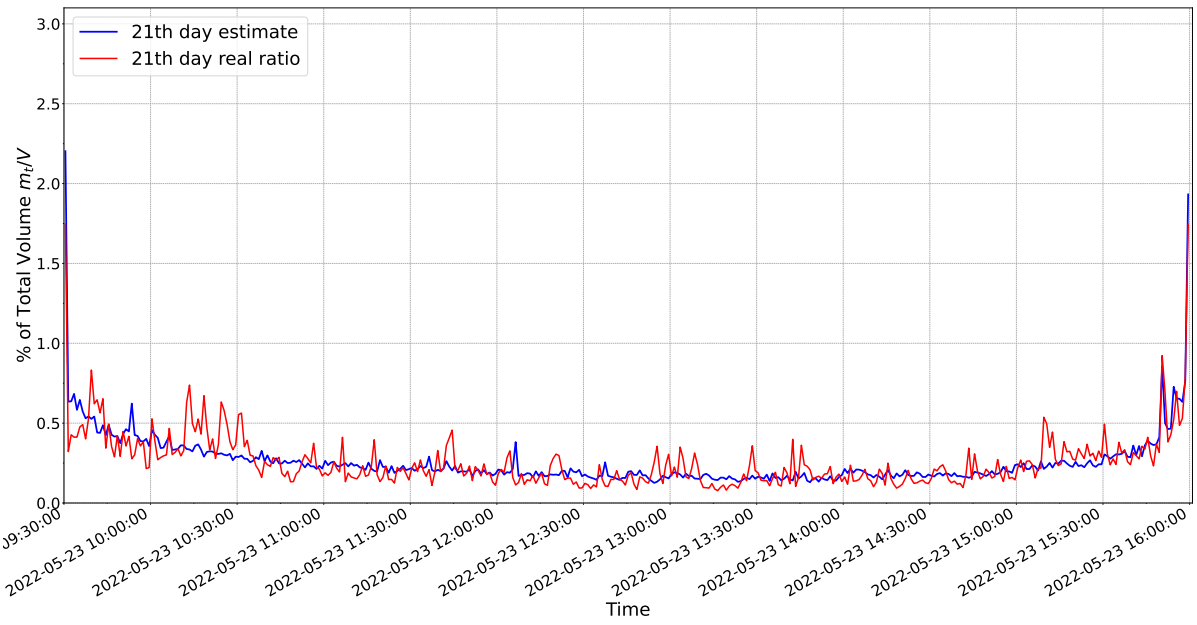

Figure 1: The average estimated m_t/V ratio versus real market ratio u_t of the 21st simulation trading day (May 23).

We observe that the plot of the solution is smoother compared to the real ratio of m_t/V of that trading day. As the window length W increases, we can expect the result to be getting better following the law of large numbers.

We instead design the convex problem in a principled way such that it overcomes the constraint described in Sec 2, *i.e.*, it will always have the optimal numerical solution, as described in Appendix A.

4.3 Dynamic Solution

Our procedure for simulating the Dynamic Solution is made up of three parts: the historical estimation of model parameters in Sec 4.3.1, and, the actual simulation of order execution in Sec 4.3.2.

4.3.1 Models estimation

In this part, we focus on the cross-validation of $b \in \mathbf{N}$ (used for empirical estimation of the covariance matrix $\hat{\sum}$). We choose it by cross-validation, reserving the first $W_{CV} = 10$ testing days of the dataset. We then compute the empirical variance of S (slippage), and choose the value of b which minimises it. Since the difference in performance of $b = 1, ..., 10$ is small after experimenting, we choose $b = 1$ to avoid overfitting. We show in Figure 3 the standard deviation of S along with the original solution.

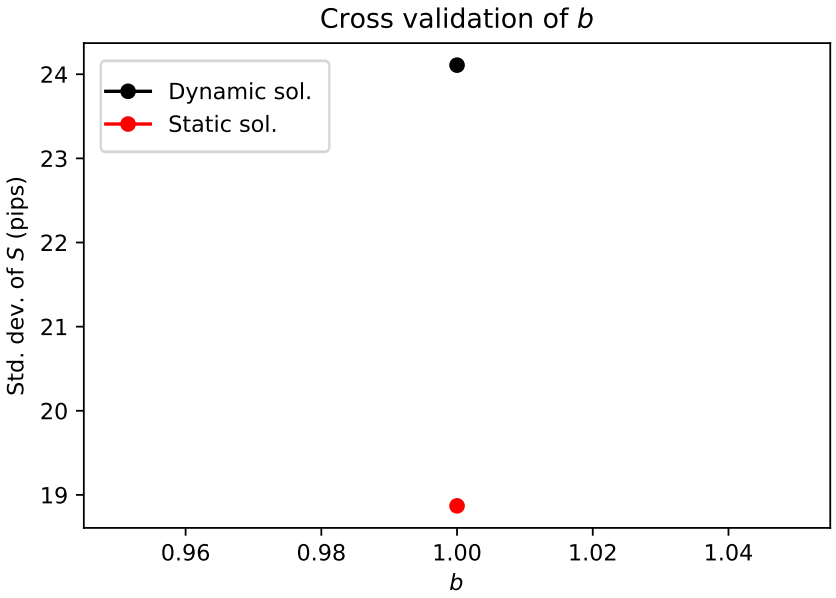

Figure 2: (Both axes are expressed in basis points).To cross validate the volume model parameter b, we compute the empirical standard deviation of S for the dynamic solution with $\lambda = \infty$, changing the value of b in the volume model. We also show the standard solution, which does not use the volume model, for comparison. For this plot we choose $b = 1$.

We show in Figure 3 the result of cumulative volume of the simulation on a sample market day, using the dynamic approach with $b = 1$ and $\lambda = 10$. We also plot the market volume m_t.

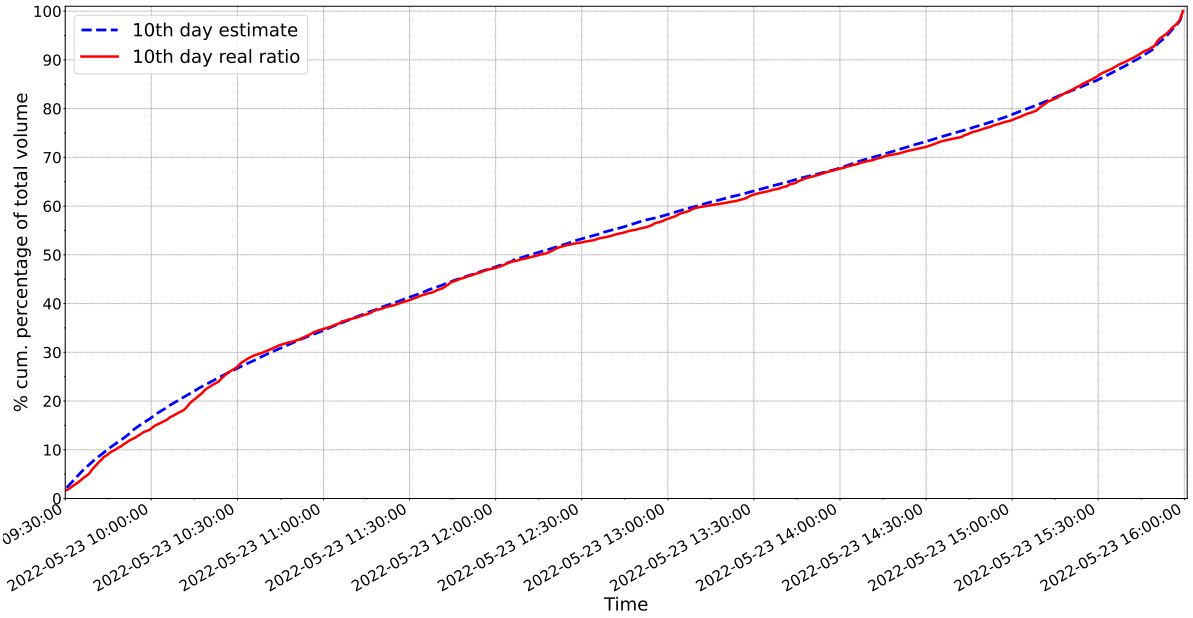

Figure 3: Simulation of order execution on the 10th simulation trading day. The cumulative plot intuitively reflects the high fidelity of the dynamic solution in VWAP tracking.

4.3.2 Implementation of SHDP solution

At the core of the implementation of dynamic programming is the updating scheme of SHDP. It connects the Multivariate Normal Distribution of the intraday volume model with the linear-quadratic stochastic control to recursively optimise each time interval from $t+1,\ldots,T$ at time t, $t=0,\ldots,T$. We present in Appendix B the implementation of SHDP which dynamically adjusts the volume model after incorporating new information m_t at time t, and the use of *backward induction* recursively to derive the optimal action u_t.

4.4 Aggregate results

We report the aggregate results from the simulation of VWAP execution on Nasdaq-100 index for a time window of 30 days (whole Dataset length $N=60$ minus $W=20$ historical time scale plus $C=10$ cross-validation time scale.) For any day $i=31,\ldots,N$, and solution method a (either the original solution or the dynamic solution for various values of λ) we obtain the simulated slippage S using

$$S^{(i,k,a)} = \frac{\sum_{t=1}^{T} p_t^{(i,k)} u_t^{(i,k,a)} - C^{(i,k)} p_{\text{VWAP}}^{(i,k)}}{C^{(i,k)} p_{\text{VWAP}}^{(i,k)}} + \sum_{t=1}^{T} \frac{s_t}{2} \left(\alpha \frac{\left(u_t^{(i,k,a)}\right)^2}{C^{(i,k)} m_t^{(i,k)}} - \frac{u_t^{(i,k,a)}}{C^{(i,k)}} \right)$$

as described in Sec 2. Note that we are simulating the transaction costs. Measuring them directly would require to actually execute $u_t^{i,a}$. Our transaction costs model is similar to the ones of other works in the literature(*e.g.* [FW13]) but involves market volumes m_t.

Then, for each solution method a define the empirical expected value of S as

$$\mathbf{E}\left[S^{(a)}\right] = \frac{\sum_{i=N-W-C+1}^{N} S^{(a)}}{N-W-C}$$

and the empirical variance

$$\mathbf{var}\left(S^{(a)}\right) = \frac{\sum_{i=N-W-C+1}^{N} (S^{(a)})^2 - \mathbf{E}\left[S^{(a)}\right]^2}{N-W-C-1}$$

In Figure 4 we show the values of these on a risk-reward plot (both axes are expressed in basis points). We observe that the dynamic approach improves over the original solution on both VWAP tracking (variance of S) and transaction costs (expected value of S), and we can select between the different behaviors by choosing different values of λ.

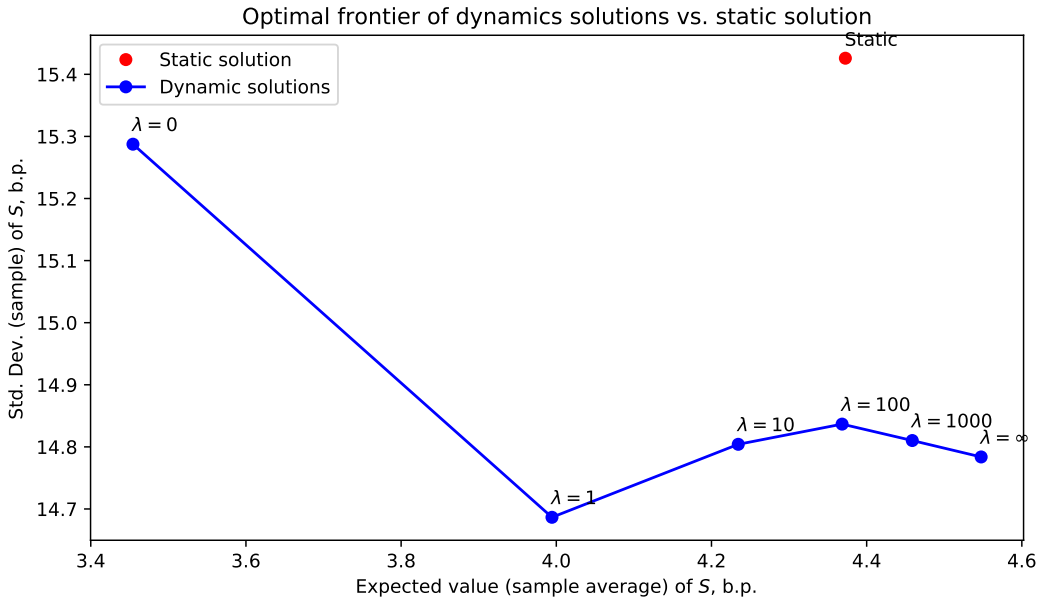

Figure 4: Risk-reward plot of the aggregate results of our simulations on real market data. Each dot represents one solution method, either the original method or the *dynamic solution* with risk-aversion parameters $\lambda = 0, 1, 10, 100, 1000, \infty$. We show the sample average of the simulated slippage, which represents the execution costs.

For the static solution the empirical average slippage is $4.37e-04$ while for the dynamic solution with $\lambda = 0$ is $3.45e-04$, about 21% lowering in execution costs. For the dynamic solution with $\lambda = 1$ the empirical average slippage is 3.99, about 8 lowering in execution costs. For the static solution the RMSE of S is $2.37e-06$ while for the dynamic solution with $\lambda = 1$ is $2.13e-06$, about 10% improvement in VWAP tracking.

4.5 time scale effects

We study the time scale effects on the empirical average slippage and RMSE of slippage of dynamic solution versus the standard solution.

In Figure 5 we show the values of empirical average slippage with time scale from 10 days to 38 days.

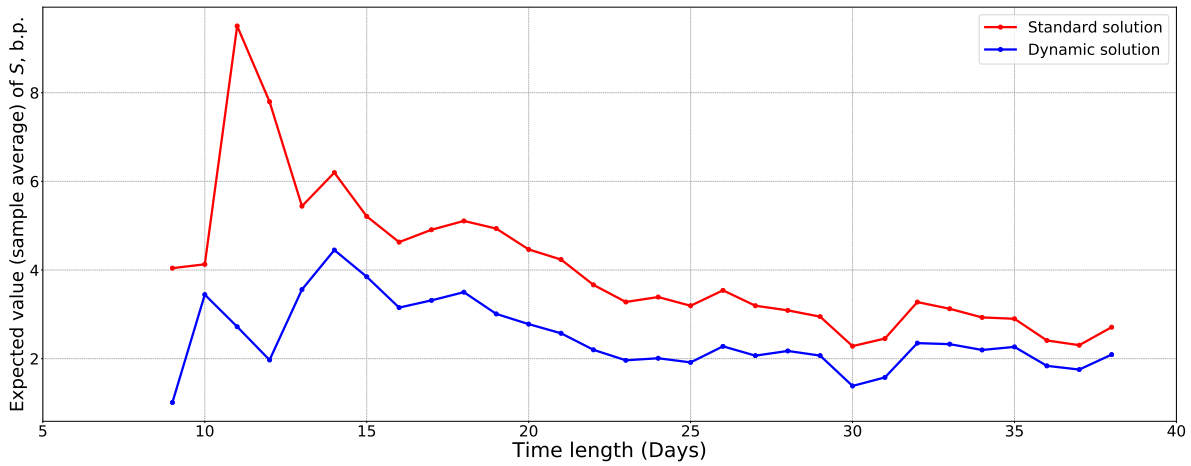

Figure 5: The empirical average slippage of standard solution versus that of dynamic solution with $\lambda = 1$.

We observe that the standard solution has larger average slippage than dynamic solution when time scale is small ($10 - 15$ days). Both average slippages of original solution and dynamic solution decay in approximately the same gradient as time scale increases ($15 - 30$ days). As time scale is large ($30 - 38$ days), the values of average slippages of dynamic and standard solution fluctuate synchronously, and the difference (average slippage of standard solution $-$ average slippage of dynamic solution) stays approximately constant.

In Figure 6 we show the values of standard deviation of S with time scale from 5 days to 38 days.

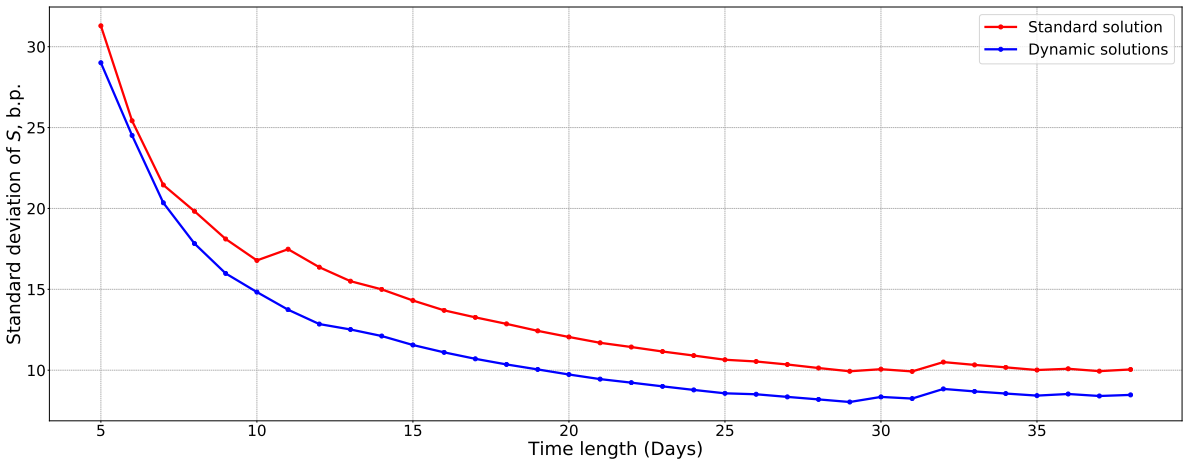

Figure 6: The standard deviation of S of the standard solution versus that of dynamic solution with $\lambda = 1$.

We observe that the standard deviations of S are approximately the same value for original solution and dynamic solution when time scale is small ($10-15$ days). Nevertheless, as time scale becomes larger ($10-25$ days) the standard deviation of S of dynamic solution

decays faster than that of original solution. From 25 days forwards, both values of standard deviations of S of standard solution and dynamic solution stop further decaying and are approximately static, with a approximately constant positive difference between them (standard solution − Dynamic solution).

5 Conclusions

We studied the problem of optimal execution under VWAP benchmark and developed a dynamic approach based on the original solution and conducted extensive comparison between them to illustrate the improvement of dynamic solution over original one.

The original solution although derived with similar assumptions to the classic [Kon02], is more flexible and can accommodate more sophisticaed models than the comparable solutions in the literature.

We build the dynamic solution based on study in [GR13], with one key addition: we update the volume model with new information under the frame of an approximate approach in dynamic programming [SBZ10], which greatly improves computational efficiency. On the one side, we improve the standard method by modelling the uncertainty of the market volumes in a principled way. On the other, we manipulate the problem to fit into the standard formalism of linear-quadratic stochastic control to avoild the computational difficulties.

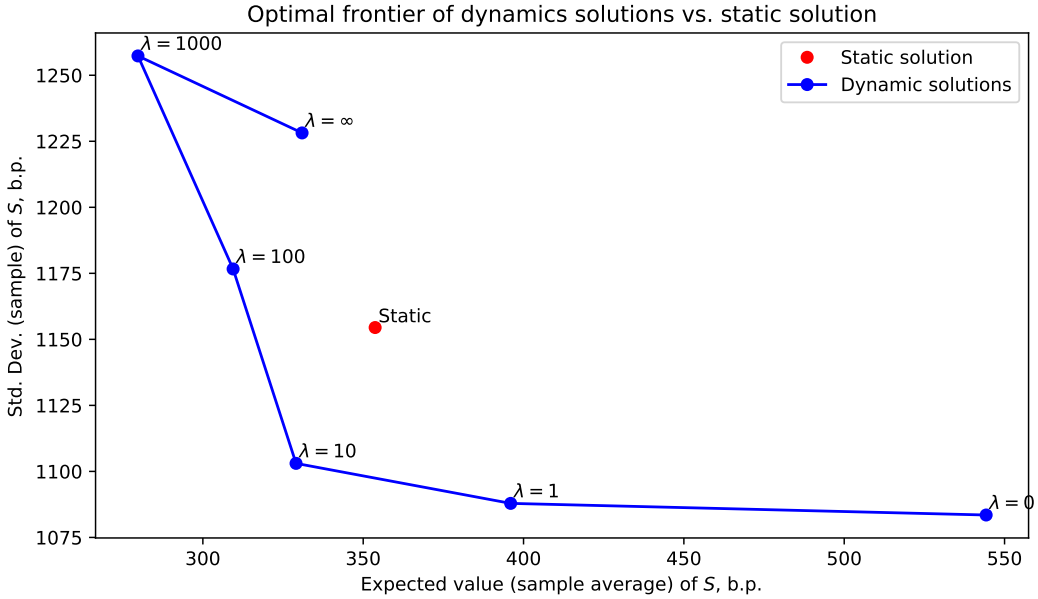

Figure 7: Simulation of order execution on a small-cap stock($<$ 30 billion) in NYSE.

The empirical result is the biggest contribution of this work. The empirical tests of Sec 4 are based on simulations with Nasdaq-100 index data designed with good statistical practices (the rolling testing ensures that all results are obtained out-of-sample). We compare the performance of the original solution, standard in the trading industry, to our

dynamic solution. With our model for market volumes (in practice a broker would have a more sophisticated market volume model) our dynamic solution improves the performance of the original solution significantly.

Our simulations quantify the improvements of our dynamic solution over the standard solution. One one side we can reduce the RMSE of VWAP by 10%, which is highly significant. On the other we can lower the execution costs by around 21%. In our test this corresponds to ~50$ of savings for an order of a million dollars (the VWAP executions are worth billions of dollars each day).

In Figure 7 we show the risk-reward plot of original solution and dynamic with data from a small-cap stock in NYSE. We raise further research topic about the factor of company capitalisation on the performance of the dynamic solution. Since all stocks in Nasdaq-100 are large-cap stocks, it's worth examining the performance of dynamic approach (with the original solution) on stocks with low price and high volatility.

Appendices

A General form of standard solution using CVXPY

In order to increase the credibility of the standard solution, we treat the problem as a standard quadratic program [BV09], and solve it efficiently using a symbolic convex optmisation suite CVXPY [DCB14], as shown in the following Python code chunk.

```
def StandardSolution(C, lambda_var, M_t, s_t, alpha_t, sigma_t):
    """Compute the standard solution given the problem parameters."""
    T = len(s_t)
    if (np.all(s_t == s_t[0])):  # constant spread
        return np.diff(np.concatenate([M_t, [1.]])) * C
    import cvxpy as cvx
    u = cvx.Variable(T)
    U = cvx.Variable(T)
    objective = cvx.square(u).T * (s_t * alpha_t / 2.) - \
      np.sign(C) * u.T * s_t / 2. + \
        lambda_var * (C**2) * (cvx.square(U).T * (sigma_t**2) - \
           2 * U.T * (sigma_t**2 * M_t))
    constraints = []
    for t in range(T):
        constraints += [U[t] == cvx.sum_entries(u[:t]) / C]
    constraints += [cvx.sum_entries(u) == C]
    constraints += [C * u >= 0]
    problem = cvx.Problem(cvx.Minimize(objective), constraints)
    problem.solve()
    return u.value.A1
```

We design the convex problem in a principled way such that it will always have the optimal numerical solution, with small computation cost, and high fidelity of tracking.

B SHDP solution

We present the implementation of SHEP scheme with stochastic affine dynamics functions f_t and stochastic quadratic cost functions g_t in the following Python code chunk.

```
def choose_action(self, u_ts, m_ts, lambda_var, prediction_result):
        """Compute u_t given the observed values
        of u_tau and m_tau for tau = 1, ..., t-1.
        Apply projection on feasible set.
        """
        self.lambda_var = lambda_var
        t = len(u_ts)
        if t == self.T - 1:
            return self.C - np.sum(u_ts)
        if lambda_var == np.inf:
            return self._choose_action_lambda_infty \
        (u_ts, m_ts, prediction_result)
        x_t = np.array([np.sum(u_ts), np.sum(m_ts)])
```

```
        pred_1_over_V = prediction_result['pred_1_over_V']
        # THIS IS A TEST
        self.Rt[t:] = self.alpha * self.s_t[t:] / 2. * \
            prediction_result['pred_1_over_mt'] / self.C
        m_t = np.concatenate(
            [m_ts, prediction_result['pred_mt']])  # for simplicity
        # I added 1/C**2 every time lambda appears to do the Stilde thing
        # final values
        self.beta_t[-1] = (self.lambda_var / (self.C**2)) * \
            self.sigma_t[-1]**2 + self.Rt[-1]
        self.gamma_t[-1] = -self.C * (self.lambda_var / (self.C**2)) *\
            (self.sigma_t[-1]**2) * pred_1_over_V
        self.delta_t[-1] = -self.rt[-1] - 2 * self.C * self.Rt[-1]
        self.l_t[-1] = self.C
        # recursion
        for tau in np.arange(self.T - 2, t - 1, -1):
            self.l_t[tau] = -(self.rt[tau] + self.delta_t[tau + 1] + \
                              2 * self.gamma_t[tau + 1] * m_t[tau]) / \
                (2 * (self.Rt[tau] + self.beta_t[tau + 1]))
            self.beta_t[tau] =
              (self.lambda_var / (self.C**2)) * self.sigma_t[tau]**2 +
              (self.Rt[tau] + self.beta_t[tau + 1])
            self.gamma_t[tau] =  -self.C * \
              (self.lambda_var /  (self.C**2)) * \
              (self.sigma_t[tau]**2) * pred_1_over_V + \
                self.Rt[tau] * self.gamma_t[tau + 1] / \
                (self.Rt[tau] + self.beta_t[tau + 1])
            self.delta_t[tau] = self.delta_t[tau + 1] + \
            2 * self.beta_t[tau + 1] * self.l_t[tau] + \
            2 * self.gamma_t[tau + 1] * m_t[tau]
        K_t = - np.array([self.beta_t[t + 1], self.gamma_t[t + 1]] \
                         ) / (self.Rt[tau] + self.beta_t[t + 1])
        # compute action
        u_t = np.dot(K_t, x_t) + self.l_t[t]
        u_t_bar = np.sign(self.C) * max(u_t * np.sign(self.C), 0)
        print K_t
        return u_t_bar
```

As shown in the above code chunk, we define the state as

$$x_t = \begin{pmatrix} \sum_{\tau=1}^{t-1} u_r \\ \sum_{\tau=1}^{t-1} m_r \end{pmatrix},$$

so that $x_1 = (0, 0)$. The action is u_t. The volume we trade during interval t,as defined in Sec 2. The disturbance is

$$w_t = \begin{pmatrix} m_t \\ V \end{pmatrix}$$

where the second element is the total market volume $V = \sum_{t=1}^{T} m_t$. For $t = 1, /dots, T$

the state trasition consists in

$$x_{t+1} = x_t + \begin{pmatrix} u_t \\ m_t \end{pmatrix}.$$

So that the dynamics matrices are

$$A_t(w_t) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \equiv \mathbf{I},$$

$$B_t(w_t) = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \equiv \mathbf{e_1},$$

$$c_t(w_t) = \begin{pmatrix} 0 \\ m_t \end{pmatrix}.$$

The objective function can be written as $\mathbf{E}_m \sum_{t=1}^{T} g_t(x_t, u_t, w_t)$ where each stage cost is given by

$$g_t(x_t, u_t, w_t) = \frac{s_t}{2}\left(\alpha \frac{u_t^2}{C m_t} - \frac{u_t}{C}\right) + \lambda \sigma_t^2 x_t^T \begin{pmatrix} \frac{1}{C^2} & -\frac{1}{CV} \\ -\frac{1}{CV} & \frac{1}{V^2} \end{pmatrix} x_t.$$

The constraint that the total executed volume is equal to C imposes the last action

$$u_T = \mu(x_T) = C - \sum_{t=1}^{T-1} u_t \equiv K_T x_t + l_t$$

with

$$K_T = -e_1^T$$
$$l_T = C.$$

This in turn fixes the value function at time T

$$v_T(x_T) = \mathbf{E}_{g_T}(x_T, K_T x_t + l_t, w_t),$$

so we can treat x_T as our final state and only consider the problem of choosing actions up to u_{T-1}.

the state trasition consists in

$$x_{t+1} = x_t + \begin{pmatrix} u_t \\ m_t \end{pmatrix}.$$

So that the dynamics matrices are

$$A_t(w_t) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \equiv \mathbf{I},$$

$$B_t(w_t) = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \equiv \mathbf{e_1},$$

$$c_t(w_t) = \begin{pmatrix} 0 \\ m_t \end{pmatrix}.$$

The objective function can be written as $\mathbf{E}_m \sum_{t=1}^{T} g_t(x_t, u_t, w_t)$ where each stage cost is given by

$$g_t(x_t, u_t, w_t) = \frac{s_t}{2}\left(\alpha \frac{u_t^2}{C m_t} - \frac{u_t}{C}\right) + \lambda \sigma_t^2 x_t^T \begin{pmatrix} \frac{1}{C^2} & -\frac{1}{CV} \\ -\frac{1}{CV} & \frac{1}{V^2} \end{pmatrix} x_t.$$

The constraint that the total executed volume is equal to C imposes the last action

$$u_T = \mu(x_T) = C - \sum_{t=1}^{T-1} u_t \equiv K_T x_t + l_t$$

with

$$K_T = -e_1^T$$
$$l_T = C.$$

This in turn fixes the value function at time T

$$v_T(x_T) = \mathbf{E}_{g_T}(x_T, K_T x_t + l_t, w_t),$$

so we can treat x_T as our final state and only consider the problem of choosing actions up to u_{T-1}.

References

[AC01] Robert Almgren and Neil Chriss. Optimal execution of portfolio transactions. *Journal of Risk*, 3:5–40, 2001.

[BDLF08] Jedrzej Białkowski, Serge Darolles, and Gaëlle Le Fol. Improving vwap strategies: A dynamic volume approach. *Journal of Banking & Finance*, 32(9):1709–1722, 2008.

[Bem06] Alberto Bemporad. Model predictive control design: New trends and tools. In *Decision and Control, 2006 45th IEEE Conference on*, pages 6678–6683. IEEE, 2006.

[Ber95] Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA, 1995.

[BFL09] J.P. Bouchaud, J.D. Farmer, and F. Lillo. *How markets slowly digest changes in supply and demand*, volume 4 of *Handbook of financial markets*. North-Holland, San Diego, CA, 2009.

[BL98] Dimitris Bertsimas and Andrew W Lo. Optimal control of execution costs. *Journal of Financial Markets*, 1(1):1–50, 1998.

[BLR12] Stephen Boyd, Sanjay Lall, and Ben Van Roy. Ee365: Stochastic control. *http://stanford.edu/class/ee365/lectures.html*, 2012.

[BMOW13] Stephen Boyd, Mark Mueller, Brendan ODonoghue, and Yang Wang. Performance bounds and suboptimal policies for multi-period investment. *Foundations and Trends in Optimization*, 1(1):1–69, 2013.

[BV09] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2009.

[DCB13] Alexander Domahidi, Eric Chu, and Stephen Boyd. Ecos: An socp solver for embedded systems. In *Control Conference (ECC), 2013 European*, pages 3071–3076. IEEE, 2013.

[DCB14] Steven Diamond, Eric Chu, and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization, version 0.2. `http://cvxpy.org/`, May 2014.

[FLM11] Jianqing Fan, Yuan Liao, and Martina Mincheva. High dimensional covariance matrix estimation in approximate factor models. *Annals of statistics*, 39(6):3320, 2011.

[FW13] Christoph Frei and Nicholas Westray. Optimal execution of a vwap order: a stochastic control approach. *Mathematical Finance*, 2013.

[GB14] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, March 2014.

[GR13] Olivier Guéant and Guillaume Royer. Vwap execution and guaranteed vwap. *arXiv preprint arXiv:1306.2832*, 2013.

[HJ11] Mark L Humphery-Jenner. Optimal vwap trading under noisy conditions. *Journal of Banking & Finance*, 35(9):2319–2329, 2011.

[TAQ] Wharton Research Data Services, TAQ Dataset. *https://wrds-web.wharton.upenn.edu/wrds/*.

[KB14] Arezou Keshavarz and Stephen Boyd. Quadratic approximate dynamic programming for input-affine systems. *International Journal of Robust and Nonlinear Control*, 24(3):432–449, 2014.

[KGM03] R. Kissell, M. Glantz, and R. Malamut. *Optimal Trading Strategies: Quantitative Approaches for Managing Market Impact and Trading Risk*. AMACOM, New York, NY, 2003.

[KH06] Wook Hyun Kwon and Soo Hee Han. *Receding horizon control: model predictive control for state models*. Springer Science & Business Media, 2006.

[Kon02] Hizuru Konishi. Optimal slice of a vwap trade. *Journal of Financial Markets*, 5(2):197–221, 2002.

[LFM03] F. Lillo, J.D. Farmer, and R.N. Mantegna. Master curve for price-impact function. *Nature*, 421(129):176–190, 2003.

[Li13] Tianhui Michael Li. *Dynamic Programming and Trade Execution*. PhD thesis, PRINCETON UNIVERSITY, 2013.

[Mad02] Ananth N Madhavan. Vwap strategies. *Trading*, 2002(1):32–39, 2002.

[MK12] James McCulloch and Vlad Kazakov. Mean variance optimal vwap trading. *Available at SSRN 1803858*, 2012.

[MS12] Ciamac C Moallemi and Mehmet Saglam. Dynamic portfolio choice with linear rebalancing rules. *Available at SSRN 2011605*, 2012.

[MWB11] Jacob Mattingley, Yang Wang, and Stephen Boyd. Receding horizon control. *Control Systems, IEEE*, 31(3):52–65, 2011.

[OW05] Anna Obizhaeva and Jiang Wang. Optimal trading strategy and supply/demand dynamics. NBER Working Papers, http://ideas.repec.org/p/nbr/nberwo/11444.html 11444, National Bureau of Economic Research, June 2005.

[Pow07] Warren B Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

[SBZ10] Joëlle Skaf, Stephen Boyd, and Assaf Zeevi. Shrinking-horizon dynamic programming. *International Journal of Robust and Nonlinear Control*, 20(17):1993–2002, 2010.