Reinforced Portfolio Optimization

Nathanael Foy

Reinforced Portfolio Optimization

Nathanaël Foy

Abstract

We consider the problem of optimal trading strategy in the presence of a price predictor, trading costs and risk control. While there exists a closed-form policy to this problem as long as precise assumptions are made on the price predictor dynamics, Reinforcement Learning allows to obtain solutions learned from historical data, without even knowing the structure of risk control and trading costs. We explore here an intermediate approach where the structure of rewards is known, but with minimalist assumptions on the price predictor dynamics. More precisely, we train an agent to trade optimally, from historical data, but without fitting any particular model on the price predictor values. This approach allows to devise a sample-efficient algorithm, compared to a generic approach where the structure of rewards is unknown.

Keywords:
Dynamic Portfolio Optimization, Reinforcement Learning, Continuous Control

1. Introduction

Optimal Trading and Portfolio Optimization have been a source of interest in both academia and industry. Harry Markowitz's seminal work [1] has laid the mathematical foundation of Portfolio Optimization. For active investors and asset managers, Portfolio Optimization has become an important industrial process. Optimizing this process even by few percents can yield significant benefits.

Portfolio Optimization aims to determine the optimal position to hold, given predictions of returns and risk control. With Dynamic Portfolio Optimization, we consider a sequence of predictions instead of a single prediction at a fixed time. The presence of trading costs makes this second problem non-trivial because the optimal policy must "aim in front of the target" as explained in [2].

In this paper, we try to solve Dynamic Portfolio Optimization using Reinforcement Learning (RL) techniques. More precisely we solve this problem with a single risky asset and under two different types of trading costs, linear and quadratic costs.

This problem has been solved analytically for both types of trading costs and we will use the closed-form policies as benchmarks. The problem with quadratic trading costs has been solved in [2] while the problem with linear trading costs has been solved in [3].

To obtain closed-form solutions, it is necessary to make precise assumptions on the price predictor dynamics. More accurately, the price predictor is assumed to be an Ornstein-Uhlenbeck process with a known decay factor that strongly determines the optimal policy. Some other work assumes different models for the price predictor dynamics [4].

In practice however, prices (and *a fortiori* price predictors) often exhibit a complex dynamics that is difficult to capture by a particular stochastic model. RL is an appealing, data-driven approach to this problem because it discards the need to assume a particular model for the predictor dynamics. The optimal solution is obtained through a training process, using historical data-points.

The first work to rigorously compare a solution learned through RL and a closed-form policy is [5]. Such a comparison is essential to assess the efficiency of an RL-based approach for a possible industrial adoption with real-world data. In [5], authors apply a generic RL algorithm called Deep Deterministic Policy Gradient (DDPG) to the problem and show that it is able to learn a close-to-optimal policy.

DDPG is generic in the sense that it can be applied to practically any problem involving a continuous control, and was not purposely designed for Dynamic Portfolio Optimization [6]. Academic Research in the field of RL has been mainly focusing on generic algorithms, and for good reasons. AphaZero is probably the most emblematic example by its ability to achieve super-human performances to possibly any two-players board game with complete information, by merely playing games against itself, and without even knowing the rules of the game at the beginning of training [7].

One major drawback when applying such algorithms to industrial problems is that such algorithms often need a large amount of data which may not always be available. While the number of games that can be generated at

Go or Chess is very large, the amount of data available (although always increasing with the advent of Big Data) to active investors and asset managers is very limited by contrast. Industry in general might be more interested by sample-efficiency than genericity, especially when data is a scarce ressource.

In this paper, we develop an RL algorithm that is specialized for Dynamic Portfolio Optimization under the same constraints considered in [5]. More precisely, we suppose that the form of risk control and trading costs are known and we use this prior knowledge to design our algorithm. We then train an agent with a limited number of synthetic data-points and compare the agent against a closed-form policy. Moreover, as in [5], no particular model is assumed for the price predictor dynamics, making this algorithm attractive for an industrial application with real-world data.

2. Description of the problem

We consider an agent that can take any position π_t, long or short, on a risky asset for all time $t = 0, ..., T$. We denote s_t the price of the risky asset at time t and the agent must pay $\pi_t s_t$ to hold the position π_t. The agent clears its position at time $t = T$ so that its gain due to holding a position π_t at time t is $\pi_t(s_T - s_t)$

We can decompose $s_T - s_t$ as a sum of a predictable component given all available information at time t, $p_t = \mathbb{E}(s_T - s_t|\mathcal{F}_t)$, and an unpredictable noise. Since the agent takes a decision at time t based on all information available at time t only (and what it can predict about s_T from that information), it is equivalent to define the agent's gain as $\pi_t p_t$, instead of $\pi_t(s_T - s_t)$.

The agent is subject to a risk penalty for holding a position π_t, $\lambda\pi_t^2$, and a cost penalty for changing its position from π_{t-1} to π_t, $C(\pi_{t-1}, \pi_t)$. We will consider two types of trading costs, linear costs $C(\pi_{t-1}, \pi_t) = \Gamma|\pi_t - \pi_{t-1}|$ and quadratic costs $C(\pi_{t-1}, \pi_t) = \frac{1}{2}\Gamma(\pi_t - \pi_{t-1})^2$.

Putting all together, the agent observes a prediction p_t, chooses a position π_t and receives a reward $\pi_t p_t - \lambda\pi_t^2 - C(\pi_{t-1}, \pi_t)$.

As evidenced by [3], the factor λ merely has a scaling impact on the sequence of optimal positions to hold and we can assume that $\lambda = \frac{1}{2}$ without loss of generality (we can rescale the sequence of positions by $\frac{1}{\sqrt{2\lambda}}$). Moreover, we assume that the observations p_t come from historical data-points that do not change under the agent's actions (the impact of the agent's actions on the price are included in the cost penalty term instead). Therefore we can rewrite our rewards as

$$R_t = -\frac{1}{2}(p_t - \pi_t)^2 - C(\pi_{t-1}, \pi_t) \tag{1}$$

The agent's goal then is to take optimal positions π_t^* so as to maximize $V_t = \mathbb{E}(\sum_{t'=t}^{t'=T} R_{t'})$.

For simplicity, we also assume that T is very large compared to all other time scales ($T \approx \infty$), so that we are only interested in a permanent regime where the time t doesn't play any role in the optimal policy.

3. Description of the algorithm

The structure of the algorithm is similar for both problems, with linear and quadratic trading costs.

First, we write path-integral equations for the optimal control variable θ^*. How to derive such equations for the control variable? Suppose that, at time s, we apply an infinitesimal perturbation on the optimal control, so that the agent outputs the control $\theta^*(s) + \delta\theta$ instead of $\theta^*(s)$ (as illustrated on Figure 1). This results in a different path and a different sequence of rewards $R'_t = R_t + \delta R_t$ for all $t \geq s$. The total value change then is $\delta V_s = \mathbb{E}(\sum_{t=s}^{t=T} \delta R_t)$. The path-integral equations for θ^* derive from the fact that, since we are at the optimum, it doesn't change much and δV must be 0 at the first order in $\delta\theta$ (i.e. $\delta V = o(\delta\theta)$, or $\frac{\partial V}{\partial \theta}(\theta^*) = 0$).

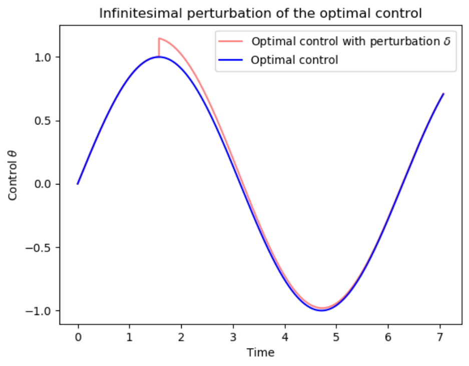

Figure 1: This figure illustrates an infinitesimal perturbation $\delta\theta$ to the optimal control at a time $s \approx 1.5$. This results in a different path and a different sequence of rewards $R'_t = R_t + \delta R_t$ for all $t \geq s$

Such equations have already been formulated in [3] in the linear case. Since we are not aware of any early work providing such equations in the quadratic case, we will simply carry out a similar analysis to obtain path-integral equations in the quadratic case as well.

Second, we use these equations to implement a fixed-point iteration that resembles a policy gradient scheme. Starting with an initial policy $\theta^{(0)}$, we run an episode, using the current policy $\theta^{(i)}$ at iteration i. Then we build a batch of training samples from the trajectory we have just simulated and update the parameters by training on that batch, to obtain $\theta^{(i+1)}$. Finally, we start a new iteration.

3.1. Quadratic costs

Suppose that we are at time t, the agent observes the prediction p_t and its current position is π_{t-1}. The optimal control is π^*_t and the reward is R_t. We prove in Appendix A that if π_{t-1} becomes $\pi_{t-1}+\delta$, then the optimal control π^*_t becomes $\pi^*_t+\alpha\delta$ where

$$\alpha = \frac{\Gamma}{1+\Gamma+k} \in [0,1[\tag{2}$$

and

$$k = \frac{-1+\sqrt{1+4\Gamma}}{2} > 0 \tag{3}$$

The corresponding reward R_t becomes

$$R'_t = R_t + \{(p_t-\pi^*_t)\alpha + (\pi^*_t - \pi_{t-1})k\}\delta + o(\delta) \tag{4}$$

Then, since $\delta V_t = \mathbb{E}(\sum_{t'=t}^{t'=T}\delta R_{t'}) = o(\delta)$ at the optimum, the optimal control is a solution of the following path-integral equation

$$\boxed{\mathbb{E}\left[(p_t-\pi^*_t)+\Gamma(\pi_{t-1}-\pi^*_t)+\sum_{t'>t}\{(p_{t'}-\pi^*_{t'})\alpha+(\pi^*_{t'}-\pi^*_{t'-1})k)\}\alpha^{t'-t-1}\right]=0} \tag{5}$$

We notice that a perturbation δ to the optimal control, at time t, results in a deviation $\delta\alpha^{t'-t}$ from the initial trajectory, at time $t' \geq t$. When $t' >> t - \frac{1}{\log\alpha}$, both trajectories merge.

We now use this equation to implement a policy gradient scheme converging towards the optimal policy.

The RL policy is implemented as a neural network taking two inputs, the current position π_{t-1} and the prediction p_t, since the optimal solution only depends on these two inputs (we have assumed that T is very large, so that we are in a permanent regime where the time t doesn't play any role in the optimal policy). In our implementation, the neural network is made of 3 dense layers.

We run an episode with our current policy $\pi^{(i)}$ (at iteration i) and we build a batch of training samples (x, y), from this trajectory, with

$$x = (\pi^{(i)}_{t-1}, p_t) \tag{6}$$

and

$$y = \frac{p_t + k\pi^{(i)}_{t+1} + (p_{t+1} - \pi^{(i)}_{t+1})\alpha + \sum_{t'>t+1}\{(p_{t'} - \pi^{(i)}_{t'})\alpha + (\pi^{(i)}_{t'} - \pi^{(i)}_{t'-1})k)\}\alpha^{t'-t-1}}{1 + \Gamma + k} \tag{7}$$

After training our neural network on this new batch of training samples, we obtain $\pi^{(i+1)}$ and then start a new iteration.

3.2. Linear costs

The structure of the optimal policy in presence of linear trading costs has been carefully analyzed in [3]. We simply summarize the results here and refer to [3] for more details.

The optimal policy consists of a no-trading zone around the predictor value, $l \leq p_t \leq u$. If the current position π_{t-1} is within the no-trading zone, i.e. $l \leq \pi_{t-1} \leq u$ then the agent must stay at this position, i.e. $\pi_t = \pi_{t-1}$. If the current position π_{t-1} is below the no-trading zone, then the agent must trade towards the lower edge of the no-trading zone, i.e $\pi_t = l$. Similarly, if the current position π_{t-1} is above the upper edge, then the agent must trade towards the upper edge, i.e. $\pi_t = u$.

The distances from the upper and lower edges, u and l, to the value of the predictor, are known to only depend on the value of the predictor. Besides, if we assume, for simplicity, that the predictor is a symmetric stochastic process with zero mean, then $u(p) = -l(-p)$, so that we only have a single control to learn, $\theta(p) = l(p)$

From [3], we know that if we change the optimal control θ^*_t by an infinitesimal perturbation δ at a time t where $\pi_{t-1} \neq \pi_t$, then the optimal position becomes $\theta^*_t + \delta$ instead of θ^*_t, for all time $t' \geq t$, until we reach a point where

$\pi_{t'-1} \neq \pi'_t$ (on the initial trajectory). At this point t', both trajectories merge and coincide afterwards.

The optimal control θ^*_t is therefore a solution of the following path-integral equation, at each time t such that $\pi_{t-1} < \pi_t$

$$\boxed{\mathbb{E}(S - L.\theta^* - 2\Gamma.\mathbb{1}_{\pi_{t'-1} > \pi_{t'}} | \pi_{t-1} < \pi_t) = 0}$$

In the above equation, L is the length of the largest time interval $I = [t, t'[$ such that $\pi_t = \pi_{t''}$ for all $t'' \in I$, S is the sum of the predictor values in I.

There exists a symmetric equation for every point where $\pi_{t-1} > \pi_t$

$$\boxed{\mathbb{E}(S + L.\theta^*(-p) + 2\Gamma.\mathbb{1}_{\pi_{t'-1} < \pi_{t'}} | \pi_{t-1} > \pi_t) = 0}$$

These equations are merely rewrites of the path-integral equations found in [3], but using an expectation symbol instead of an integral symbol.

Now let's describe our policy gradient scheme. Since the optimal control θ^* only depends on the predictor value p, the policy is implemented as a neural network with a single input p.

We run an episode with our current policy $\pi^{(i)}$ (at iteration i) and we build a batch training samples.

As the path-integral equations are only valid at times t such that $\pi_{t-1} \neq \pi_t$, we only consider such events for training. Besides, because of the presence of the factor L, we need to weight each sample by a weight $\omega = L$. We thus build a batch of weighted training samples (x, y, ω) for all time t where $\pi_{t-1} \neq \pi_t$, with

$$x = \begin{cases} p_t & \text{if } \pi_{t-1} < \pi_t \\ -p_t & \text{if } \pi_{t-1} > \pi_t \end{cases} \tag{8}$$

$$y = \begin{cases} \frac{S - 2\Gamma \mathbb{1}_{\pi_{t'-1} > \pi_{t'}}}{L} & \text{if } \pi_{t-1} < \pi_t \\ \\ \frac{-S - 2\Gamma \mathbb{1}_{\pi_{t'-1} < \pi_{t'}}}{L} & \text{if } \pi_{t-1} > \pi_t \end{cases} \tag{9}$$

and

$$\omega = L \tag{10}$$

Here is an important caveat: were the training samples not weighted by ω, the policy would not converge towards the optimum. Think about the simple case where the predictor can take two values 1 and -1, and the predictor takes the opposite value with probability p and keeps the same value as the previous one with probability $1-p$. The optimal control is $\theta^*(p) = 1 - 2\Gamma p$ (assuming $\Gamma < \frac{1}{2}$). If the training samples are not weighted by L, then the policy converges towards $1 + 2\Gamma\frac{p\log p}{1-p} \neq 1 - 2\Gamma p$.

4. Numerical results

In this section, we train an agent with a limited number of synthetic data-points and compare the agent against a closed-form policy. The data-points are generated with a discrete Ornstein-Uhlenbeck process

$$p_t = \rho.p_{t-1} + \beta.\xi_t \tag{11}$$

where ξ_t represents a noise with Gaussian distribution ($\mathcal{N}(0,1)$).

The parameters ρ and β of the Ornstein-Uhlenbeck process are unknown to the RL algorithm that doesn't assume any model for the data-points. Only Γ is known to the agent (the model for trading costs is always known as soon as one needs to backtest policies with historical data). Therefore, the interest of the RL policy resides in the possibility to be trained with data-points for which we don't assume any model. This can be beneficial when working with real-world data.

4.1. Quadratic costs

A closed-form policy for the quadratic case can be found in [2] and is a linear combination of the current position π_{t-1} and the predictor value p_t

$$\pi^*(\pi, p) = (1-\omega)\pi + \omega\psi p \tag{12}$$

with $\psi = \frac{\omega}{1+(\omega-1)\rho}$, $\omega = \frac{-1+\sqrt{1+4\Gamma}}{2\Gamma}$.

The numerical results have been obtained with $1-\rho = 0.1$, $\beta = 0.1$, and $\Gamma = 1$.

1000 synthetic data-points have been generated and re-used for every iteration.

Figure 2 represents the learning curve. Convergence is reached after about 40 iterations.

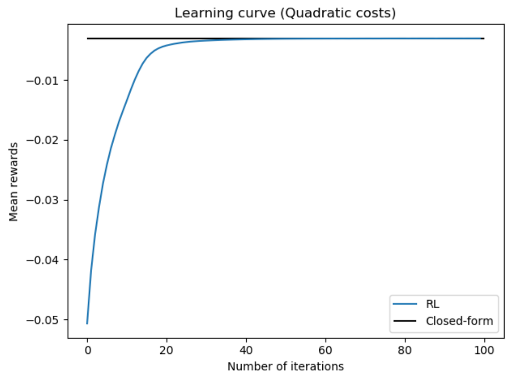

Figure 2: Learning curve representing the evolution of mean rewards obtained by the agent in the course of training.

Figure 3 represents the function $x \to \pi^*(0.1, x)$ for both the closed-form policy and the RL policy (after 100 iterations). The agent we obtain is close to the closed-form policy. A discrepancy can be observed for values of the predictor below -0.1. This can be explained by the fact that, when the current position is 0.1, predictor values below -0.1 are less observed (because of the predictor auto-correlation) and, as a consequence, training is less accurate in this region.

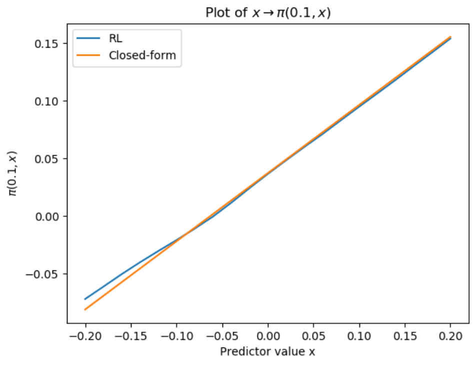

Figure 3: Plot of function $x \to \pi^*(0.1, x)$ for the closed-form policy and RL policy.

4.2. Linear costs

A closed-form policy in the linear case can be found in [3]. It is worth noting that this closed-form policy is optimal only in the continuous limit

$\beta << \Gamma$. Therefore, we choose the following parameters $1-\rho = 0.01$, $\beta = 0.01$ and $\Gamma = 3$, to make a fair comparison.

The closed-form policy is given by

$$\theta(p) = \frac{e^{-ap^2} - e^{-aq^2}}{2aJ} \tag{13}$$

where q is given as a function of p by

$$p - q = 2\Gamma(1-\rho) - Ie^{ap^2} + \frac{K(e^{-ap^2} - e^{-aq^2})}{J} \tag{14}$$

with

$$a = \frac{1-\rho}{\beta^2},\ I = \int_p^q e^{ax^2}dx,\ I = \int_p^q e^{-ax^2}dx,\ K = \int_p^q \int_{y=p}^{y=x} e^{a(x^2-y^2)}dxdy \tag{15}$$

Since only a subset of events are used to build training batches (events such that $\pi_t \neq \pi_{t-1}$), the algorithm needs more data to train. Moreover, episodes must be long enough so that we have at least several such events. This is especially true when parameters $1-\rho$ and β are small because we are closer to the continuous time limit. Therefore we have generated 100,000 data-points at each iteration to train the agent, and used a constant band policy $\theta^{(0)}(p) = p - \theta_0$ as the initial policy (where θ_0 is determined through a coarse-grained grid search), instead of a completely random one to avoid cases where the policy doesn't generate any event $\pi_t \neq \pi_{t-1}$.

Figure 4 represents the learning curve. Convergence is reached after about 10 iterations.

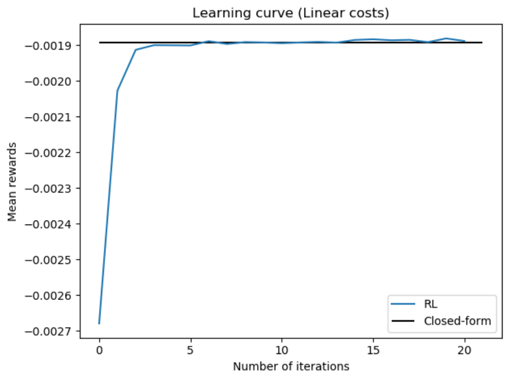

Figure 4: Learning curve representing the evolution of mean rewards obtained by the agent in the course of training. It should not be much a surprise that the RL policy is able to obtain slightly better results than the closed-form policy, because the latter is optimal only in the continuous time limit whereas we're doing a benchmark on a discrete Ornstein-Uhlenbeck process (even if the parameters $1 - \rho$ and β have been chosen so as to be close to the continuous time limit).

Figure 5 represents the function $p \to p - \theta(p)$ for both the closed-form policy and the RL policy (after 20 iterations). We represent the two functions in a region where there are a significant number of training samples.

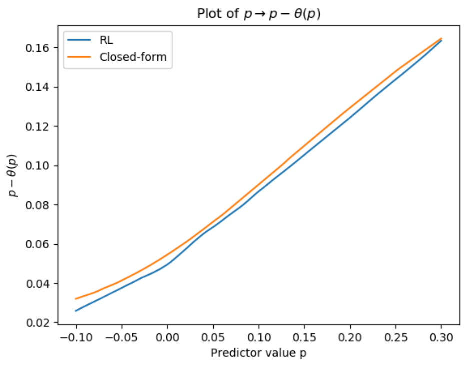

Figure 5: Plot of function $p \to p - \theta(p)$ for the closed-form policy and RL policy.

Conclusion

We have introduced two new RL algorithms to solve the problem of Dynamic Portfolio Optimization, with linear and quadratic trading costs. We have shown that these algorithms were able to learn the optimal policy using a limited number of training samples. The RL policies have been compared against closed-form policies on synthetic data.

Some further steps could be to apply these algorithms to real-world data and compare the results with other methods. Besides, we have only considered a simple setup with a single risky asset. It could be interesting to extend this approach to a more complex setup with multiple risky assets.

At last, we believe that using the differentiable structure of rewards to obtain path-integral equations for the optimal control, and using these equations to implement a policy gradient scheme, is a general approach that could be applied to other continuous control problems beyond Portfolio Optimization.

Appendix A. Perturbation on the optimal path (quadratic costs environment)

We prove here the following proposition (in an environment involving quadratic trading costs) that describes how the optimal control deviates from the initial one following an infinitesimal perturbation δ to the current position π_{t-1} at time t.

Proposition 1. *Suppose that we are at time t, the agent observes the prediction p_t and its current position is π_{t-1}. The optimal control is π_t^* and the reward is $R_t = -\frac{1}{2}(p_t - \pi_t^*)^2 - \frac{1}{2}\Gamma(\pi_t^* - \pi_{t-1})$. If π_{t-1} becomes $\pi_{t-1} + \delta$, then the optimal control π_t^* becomes $\pi_t^* + \alpha\delta$ where*

$$\alpha = \frac{\Gamma}{1 + \Gamma + k} \in [0, 1[\tag{A.1}$$

and

$$k = \frac{-1 + \sqrt{1 + 4\Gamma}}{2} > 0 \tag{A.2}$$

The corresponding reward R_t becomes

$$R_t' = R_t + \{(p_t - \pi_t^*)\alpha + (\pi_t^* - \pi_{t-1})k\}\delta + o(\delta) \tag{A.3}$$

Proof. We first prove by induction that the value function $V_t = \max_{\pi_t}\{\mathbb{E}(\sum_{t' \geq t} R_{t'})\}$ is of the form

$$V_t = \max_{\pi_t}\{-\frac{1}{2}(\pi_t - p_t)^2 - \frac{1}{2}\Gamma(\pi_t - \pi_{t-1})^2 - \frac{1}{2}k_t(\pi_t - a_t)^2 - b_t)\} \tag{A.4}$$

where a_t and b_t are some quantities that do not depend on π_t, and k_t is defined by the recursive formula:

$$k_T = 0, \text{ and } k_{t-1} = \frac{(1+k_t)\Gamma}{1+\Gamma+k_t} \tag{A.5}$$

The above statement is true when $t = T$ because

$$V_T = \max_{\pi_T}\{-\frac{1}{2}(\pi_T - p_t)^2 - \frac{1}{2}\Gamma(\pi_T - \pi_{T-1})^2\} \tag{A.6}$$

Assuming that this is true for some t, let's prove it for $t-1$. Since

$$V_t = \max_{\pi_t}\{-\frac{1}{2}(\pi_t - p_t)^2 - \frac{1}{2}\Gamma(\pi_t - \pi_{t-1})^2 - \frac{1}{2}k_t(\pi_t - a_t)^2 - b_t)\} \tag{A.7}$$

we have

$$\pi_t^* = \frac{p_t + \Gamma\pi_{t-1} + k_t a_t}{1+\Gamma+k_t} \tag{A.8}$$

As a consequence, V_t is a quadratic expression in the variable π_{t-1}, i.e. $V_t = A.\pi_{t-1}^2 + B.\pi_{t-1} + C$, where

$$A = -\frac{1}{2}\frac{\Gamma^2 + \Gamma(1+k_t)^2 + k_t\Gamma^2}{(1+\Gamma+k_t)^2} = -\frac{1}{2}\frac{(1+k_t)\Gamma}{1+\Gamma+k_t} = -\frac{1}{2}k_{t-1} \tag{A.9}$$

On the other hand, we have

$$V_{t-1} = \max_{\pi_{t-1}}\{-\frac{1}{2}(\pi_{t-1} - p_{t-1})^2 - \frac{1}{2}\Gamma(\pi_{t-1} - \pi_{t-2})^2 + \mathbb{E}(V_t)\} \tag{A.10}$$

Since A doesn't depend on p_t, we can write that

$$\mathbb{E}(V_t) = -\frac{1}{2}k_{t-1}(\pi_{t-1} - a_{t-1})^2 - b_{t-1} \tag{A.11}$$

where a_{t-1} and b_{t-1} are some quantites independent from π_{t-1}.

It follows that

$$V_{t-1} = \max_{\pi_{t-1}}\{-\frac{1}{2}(\pi_{t-1} - p_{t-1})^2 - \frac{1}{2}\Gamma(\pi_{t-1} - \pi_{t-2})^2 - \frac{1}{2}k_{t-1}(\pi_{t-1} - a_{t-1})^2 - b_{t-1}\} \tag{A.12}$$

thus finishing the proof by induction.

The proposition then results from the fact that if x is the optimum of some quadratic expression $Ax^2 + Bx^2 + C$, and if B is changed into $B + \delta B$ then the optimum x becomes $x' = x - \frac{\delta B}{2A}$.

Finally, in the permanent regime, we have

$$k = k_\infty = \frac{-1 + \sqrt{1 + 4\Gamma}}{2} \tag{A.13}$$

□

References

[1] H. Markowitz, Portfolio selection, 2nd Edition, no. 16 in Monograph / Cowles Foundation for Research in Economics at Yale University, Yale Univ. Press, New Haven, Conn. [u.a.], 1970.

[2] N. Grleanu, L. H. Pedersen, Dynamic trading with predictable returns and transaction costs, The Journal of Finance 68 (6) (2013) 2309–2340. doi:10.1111/jofi.12080.

[3] J. de Lataillade, A. Chaouki, Equations and shape of the optimal band strategy (2020). arXiv:2003.04646.

[4] J. Muhle-Karbe, M. Reppen, H. M. Soner, A primer on portfolio choice with small transaction costs (2016). arXiv:1612.01302.

[5] A. Chaouki, S. Hardiman, C. Schmidt, E. Sri, J. Lataillade, Deep deterministic portfolio optimization, The Journal of Finance and Data Science (07 2020). doi:10.1016/j.jfds.2020.06.002.

[6] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning., in: ICLR (Poster), 2016.

[7] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, D. Hassabis, A general reinforcement learning algorithm that masters chess, shogi, and go through self-play, Science 362 (6419) (2018) 1140–1144. doi:10.1126/science.aar6404.