QF209 Machine Learning in Quantitative Finance

AY 2024-2025, Term 1

Derivative Hedging using RL

Section G1
Group 2

Chen Jia Jie (01431006)

Goh Wei Han (01431924)

Lee Yin Weng (01470823)

Lim Yi Xin (01409539)

New Wee Hian Ryan (01470668)

Content Page

Executive Summary **3**
1.0 Introduction to Research Problem and its Impact **4**
2.0 Overview of RL **5**
3.0 Implemented Models **6**
3.1 Baseline Model: Delta Hedging 6
3.2 Deep Q-Networks (DQN) 6
3.2.1 DQN Model Architecture 7
3.3 Deep Deterministic Policy Gradient (DDPG) 8
3.3.1 DDPG Model Architecture 8
4.0 Data Simulation and Environment Setup **9**
4.1 Getting Data 9
4.2 State Space 10
4.3 Reward Function 11
4.4 Action Space 12
5.0 Training and Testing Procedures **12**
5.1 Training the Models 12
5.2 Testing the Models 13
5.3 Code Implementation 13
6.0 Evaluation Metrics **14**
7.0 Experimental Results and Analysis **14**
7.1 Cases of Varying Non-zero Transaction Costs 15
7.2 Cases of Hedging Options with Varying Maturities 16
7.3 Modifying the State Space 17
7.3.1 Analysis of Removing Delta from State Space 18
7.3.2 Analysis of Removing Option Price from State Space 19
7.3.3 Analysis of Removing Other Components in Features DataFrame 20
8.0 Conclusion and Future Directions **23**
Appendices **24**
Appendix 1 24
Appendix 2 25
Appendix 3 26
Appendix 4 27

Executive Summary

Financial derivatives trading is complex, with significant risks due to fluctuations in the prices of underlying assets. Traditional hedging strategies like Delta Hedging, based on the Black-Scholes-Merton model, aim to maintain a delta-neutral portfolio to mitigate sensitivity to price movements. However, these strategies presume constant volatility, zero transaction costs, and perfect liquidity—assumptions rarely met in real-world markets, limiting their effectiveness in managing nuanced risks.

Our approach uses Reinforcement Learning (RL) to create automated, data-driven hedging strategies for a position of 100 European call options on Apple Inc., using only Apple shares as the hedging instrument. This method incorporates actual historical data and estimates volatility dynamically through a 30-day rolling standard deviation of returns, closely simulating real market conditions.

We compare several models: a Baseline Model of Delta Hedging, serving as a benchmark by adjusting stock positions daily based on delta values without adaptive learning; Deep Q-Networks (DQN), enhancing classical Q-learning with deep neural networks to handle high-dimensional state spaces for optimal action selection; and Deep Deterministic Policy Gradient (DDPG), suitable for continuous action spaces and integrating policy gradient methods with Q-learning in an Actor-Critic framework for precise portfolio management.

The state space features include time to expiry, stock price, number of stocks held, cash balance, and a comprehensive set of financial indicators and option Greeks. The reward function is meticulously designed to account for stock and option price changes, transaction costs, and penalises portfolio volatility to promote stable returns.

Key findings reveal that as transaction costs increase, both DQN and DDPG begin to surpass the Delta Hedging strategy, with DQN displaying remarkable adaptability by maintaining lower P&L variability in high-cost environments. DDPG excels in managing short to medium-term maturities (30 to 60 days), showing lower P&L volatility, whereas traditional Delta Hedging outperforms RL models for longer maturities (90 to 120 days), suggesting a need for further enhancements in RL models for extended durations. Removing delta from the state space significantly raises P&L volatility for all models, emphasising its importance in hedging decisions, while excluding the option price leads to improved stability in RL models, indicating that a focus on fundamental market variables can enhance robustness.

In conclusion, RL models, particularly DDPG, provide superior hedging strategies under certain conditions, effectively managing risk. Future research should explore advanced RL algorithms, enhanced model architectures, dynamic feature selection, and agent stacking to potentially develop more robust hedging strategies. This research contributes to more resilient and effective risk management tools, capable of navigating real-world financial market complexities.

1.0 Introduction to Research Problem and its Impact

The field of financial derivatives trading is marked by its complexity and the significant risks associated with the rapid fluctuations in the prices of underlying assets. Traditional hedging strategies, such as Delta Hedging, have been utilised to manage the risks associated with these price movements. Delta Hedging, based on the Black-Scholes-Merton (BSM) model, seeks to maintain a delta-neutral portfolio, aiming to neutralise the portfolio's sensitivity to price movements in the underlying asset. This strategy typically requires continuous rebalancing of the portfolio, assuming stable market conditions—constant volatility, zero transaction costs, and perfect liquidity—a set of assumptions that rarely holds true in the turbulent real-world markets. These assumptions limit the applicability of Delta Hedging in managing the nuanced risks presented by financial markets characterised by high transaction costs and volatility.

The advent of reinforcement learning (RL) in financial hedging introduces a transformative approach that overcomes many limitations of traditional methods. RL, a dynamic branch of machine learning, enables agents to learn optimal strategies through systematic interaction with a constantly changing environment, maximising cumulative rewards without relying on preconceived models. Unlike traditional methods, RL adapts to real-time market dynamics, continuously updating hedging positions based on live market data and managing transaction costs effectively. This adaptability makes RL ideal for the volatile nature of financial markets, providing a flexible and robust framework for risk management.

Our research utilises RL to devise automated, data-driven hedging strategies that can dynamically adjust to real-world market conditions. Specifically, we focus on a practical trading scenario where an RL agent manages a position of 100 European call options on Apple, using only Apple shares as the hedging instrument from the initiation of the trade at time $t = 0$ until the options' expiry. The agent's strategy involves adjusting the holdings of Apple shares in response to changes in market conditions, thereby managing the risk associated with the option's position through the strategy's duration.

This approach leverages actual historical data and market volatility to simulate and price options, which is a departure from existing research that often rely on constant volatility assumptions. By applying RL, this research not only addresses the deficiencies of conventional hedging techniques but also enhances the precision and effectiveness of hedging practices. It paves the way for broader applications of RL across various financial instruments and conditions, enhancing the resilience and efficiency of financial markets.

Thus, this study aims to demonstrate the practical impacts and potential of RL in transforming financial risk management by providing a detailed analysis of RL-based strategies against traditional Delta Hedging, showcasing the advantages of incorporating real-time data and learning algorithms in crafting superior hedging strategies.

2.0 Overview of RL

RL (RL) provides a robust framework for solving multi-period optimal control problems, particularly in dynamic environments such as financial markets. At its core, RL utilises the Markov Decision Process (MDP), which details the sequential decision-making process through states and actions, guided by a policy. At each time step t, the agent selects an action a_t, influenced by the current state s_t and the policy π. The environment then yields a reward r_t and transitions to the next state s_{t+1}, with transitions modelled by probabilities $p(s_{t+1} | s_t, a_t)$. The aim is to discover the optimal policy π^* that maximises the cumulative discounted rewards:

$$R_0 = \sum_{t=0}^{T} \gamma^t r_t(s_t, a_t, s_{t+1})$$

where γ is the discount factor that weighs the importance of immediate versus future rewards.

The key to solving this optimization is the action-value function $q^*(s, a)$, which is determined by the Bellman optimality equation:

$$q_t^*(s, a) = \mathbb{E}[r_t + \gamma max_{a'} q_{t+1}^*(s', a') | s_t = s, a_t = a]$$

This equation recursively relates the value of a state-action pair to the expected return of optimal actions taken thereafter. In hedging, this translates to dynamically adjusting positions in derivatives like options to manage financial risks effectively. The RL agent iteratively updates its strategy based on observed market conditions, improving its hedging actions over time to minimise cost and risk, while adapting to the evolving financial landscape.

3.0 Implemented Models

We implemented a range of models to address the challenges of financial hedging, varying from traditional financial models to advanced RL algorithms. Figure 1 provides an overarching schematic that details the setup for both the DQN and DDPG models, illustrating their inputs, neural network architectures, and how they interact with their environment to determine actions.

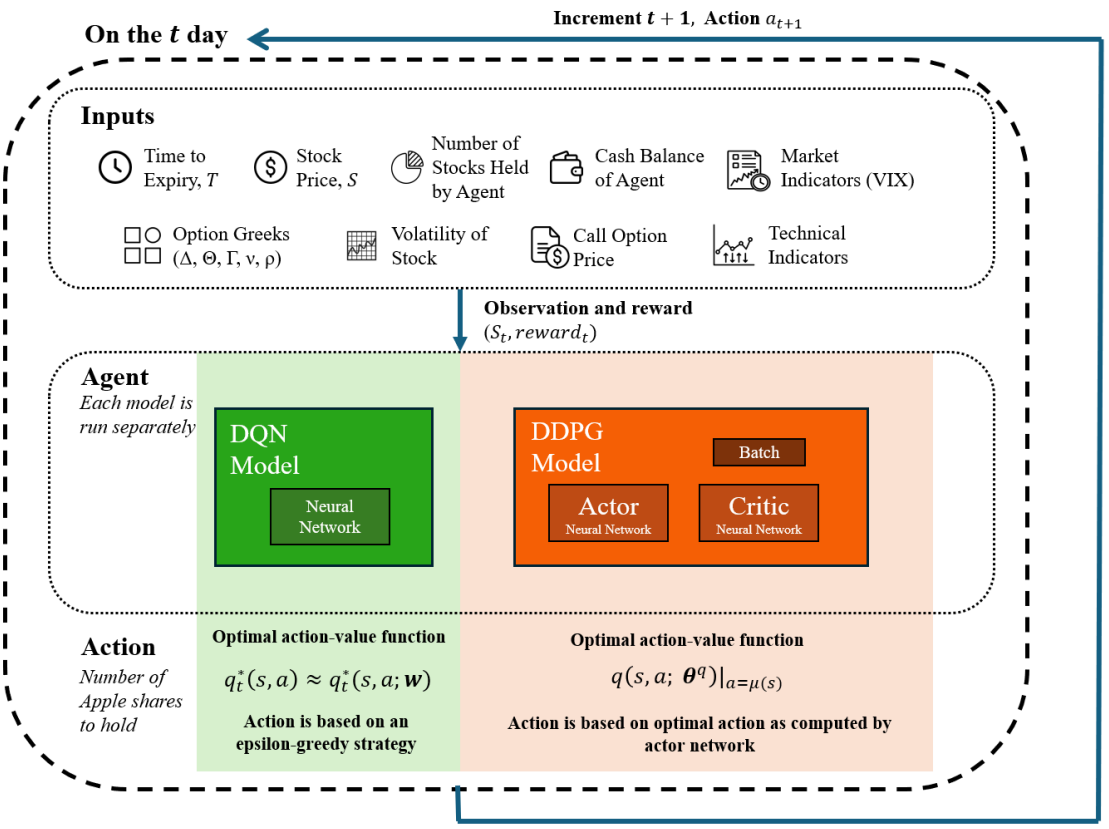


Figure 1: Overall Schematic for DQN and DDPG Model

3.1 Baseline Model: Delta Hedging

Our baseline model employs Delta Hedging, a traditional approach in financial trading. It adjusts stock positions daily according to the delta value, which measures an option's price sensitivity to changes in its underlying asset. This is not a RL model as it does not involve learning from historical data or adapting based on previous outcomes. This simply serves as a benchmark against which the performance of our more complex RL models is measured.

3.2 Deep Q-Networks (DQN)

The first of our RL models is the Deep Q-Network (DQN). This model enhances the classical Q-learning framework by integrating deep neural networks to handle high-dimensional state spaces typically encountered in financial markets. DQN uses a series of layers and nonlinear

transformations to approximate the Q-function, which estimates the reward expected from taking a certain action in a given state. This capability allows the model to discern and enact the most rewarding actions based on the learned values.

3.2.1 DQN Model Architecture

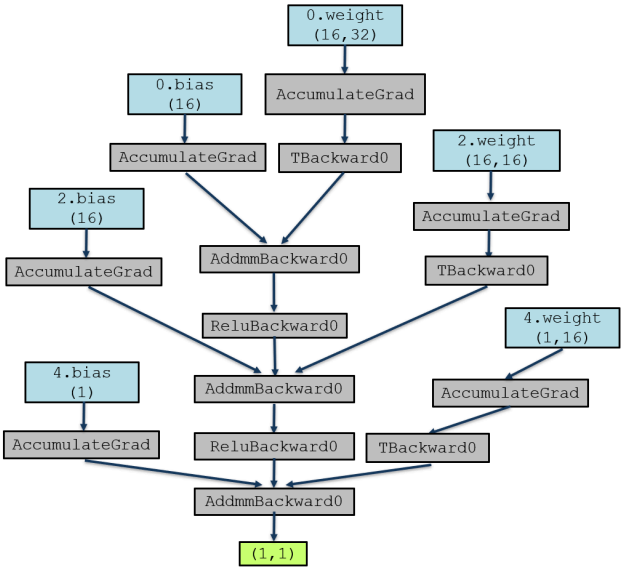


Figure 2: Overview Schematic of DQN Neural Network

Figure 2 illustrates the architecture of the deep neural network used in DQN. The diagram highlights the components of the first hidden layer, including 0.weight and 0.bias, which represent the weights and biases, respectively. The weights are arranged in a matrix format (16, 32), indicating that the network processes an input of 32 features that represent the state and converts these into 16 outputs. The number 32 can be adjusted as features are added to or removed from the state representation. The network consists of two hidden layers; following each linear transformation within these layers, a ReLU function is applied to introduce nonlinearity into the processing pathway. The outputs from the final hidden layer, 16 in total, are then consolidated into a single scalar output in the last layer, depicted as (1,1) in size.

The grey boxes labelled AccumulateGrad, TBackward0, and AddmmBackward0 denote components of the backward pass. In this phase, the network computes gradients for each weight and bias based on the error signal—a key step in training. These gradients are critical for iteratively refining the network's predictions, enhancing the model's ability to determine the most effective actions based on its evolving understanding of the state space.

3.3 Deep Deterministic Policy Gradient (DDPG)

Our second RL model, the Deep Deterministic Policy Gradient (DDPG), is designed for complex environments with continuous action spaces—typical of financial markets where actions such as trading volumes can vary continuously. DDPG effectively integrates the advantages of policy gradient methods with the predictive stability of Q-learning, enabling precise management of dynamic portfolios that necessitate nuanced and ongoing adjustments to hedge positions.

DDPG employs an Actor-Critic framework to optimise decision-making processes. The Actor network proposes the optimal action for any given market state, while the Critic network assesses the value of the state-action pair, thus providing essential feedback to the Actor. This continuous loop of action and evaluation facilitates the refinement of strategies, allowing the system to adapt and optimise actions based on learned outcomes and their effectiveness.

3.3.1 DDPG Model Architecture

Actor Network: The architecture of the Actor network in DDPG comprises three hidden layers, each containing 12 neurons, with ReLU activation functions leading into a single output layer that determines the action to be taken (see Figure 3). This setup ensures that the Actor can process the state inputs into actions that are finely tuned for the dynamics of the market environment.

Critic Network: In contrast, the Critic network features a more robust configuration with three hidden layers as well, but each containing 24 neurons (see Figure 4). This enhanced complexity allows the Critic to more effectively learn and evaluate the nuances of different state-action pairs. The larger network capacity is critical for accurately estimating the value function, thereby providing more precise feedback to the Actor. This arrangement helps in refining the Actor's policy by offering detailed insights into the consequences of actions taken, leading to more informed and effective decision-making processes in portfolio management.

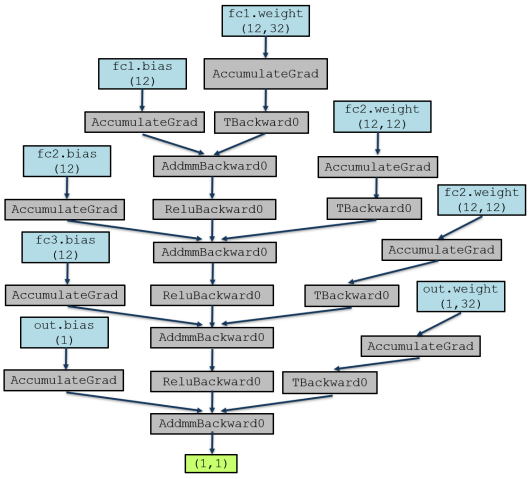


Figure 3: Overview Schematic of Actor Neural Network for DDPG Model

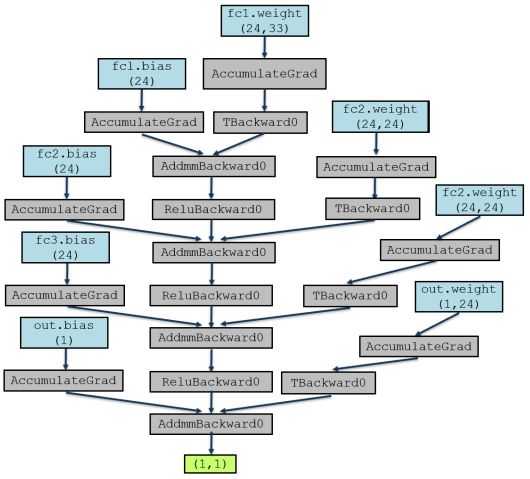


Figure 4: Overview Schematic of Critic Neural Network for DDPG Model

4.0 Data Simulation and Environment Setup

4.1 Getting Data

In our research focusing on hedging strategies involving Apple shares, we utilised two distinct types of data: historical stock prices and options data. The historical stock prices of Apple were readily available and downloaded via yfinance, which provided a robust dataset for our simulations. For the options data, we simulated call option data derived from the same historical stock prices.

The use of simulated data was driven by the common challenge of accessing comprehensive options data sources, which often require expensive subscriptions, such as those provided by Wharton Research Data Services. Employing the Black-Scholes model for simulation served as a strategic choice to create market-realistic options data, thereby offering a viable alternative for users without access to premium data services (see Appendix 1). For those who have access to full options data, our code can be easily adapted to incorporate real data.

To simulate realistic options data, we utilised the Black-Scholes model with actual historical stock prices from yfinance. Unlike existing research that often relies on fixed volatility values, we dynamically estimated volatility using a 30-day rolling standard deviation of returns. This approach better captures the true market conditions and allows our RL models to train in an environment that closely mirrors actual market dynamics. In our simulations, we standardised key parameters such as the strike price (K) and time to maturity (T). The risk-free rate was set using the 3-month US Treasury rate, maintaining alignment with standard financial modelling practices.

4.2 State Space

The design of the state space in RL is crucial as it forms the foundation for how the agent observes the environment and makes decisions. This corresponds to the inputs as depicted in Figure 1. Our state space is carefully tailored with core features that are essential for optimising our hedging strategies. These features include Time to Expiry (T) of the options, which influences the urgency and tactical approach; the Stock Price (S), critical for the real-time valuation of options; the Number of Stocks (num_stk), which reflects the quantity of shares the agent holds, affecting market exposure; and notably, the Cash Balance. The inclusion of Cash Balance is vital for monitoring the agent's financial status and maintaining a self-financing operation. This self-financing aspect is particularly important as it ensures the hedging strategy operates without the need for additional external funding. This would also be helpful for traders who face regulatory or organisational constraints that limit external capital infusion.

To enhance the state space, we have developed a Features DataFrame that aggregates a comprehensive set of financial indicators, thus enriching the agent's view of market dynamics. This DataFrame includes risk-free rates, employing 3-month Treasury rates for a realistic benchmark of these rates, along with log returns which provide a normalised measure of stock price movements.

In addition to basic market data, the DataFrame includes detailed option-related data such as volatility and call option prices, which are fundamental to the real-time recalibration of the hedging strategy according to current market volatility. Market indicators, including the VIX index, are also integrated to mirror the market's volatility expectations, enhancing the model's ability to dynamically assess risks.

Technical indicators included in the DataFrame utilise a mixture of 10, 30, and 200-day rolling windows to track and predict price movements. These indicators include Moving Averages (MAs), Exponential Moving Averages (EMAs), Rate of Change (ROC), Momentum (MOM), Relative Strength Index (RSI), and Stochastic Oscillators (%K, %D). While generally considered lagging indicators that help forecast future market trends based on historical data, it is notable that stochastic oscillators can serve as leading indicators, providing early warnings of potential market shifts.

Moreover, the incorporation of option Greeks such as Delta, Gamma, Theta, Vega, and Rho into the DataFrame is critical. These metrics are indispensable for understanding and managing the sensitivities of option prices to various market factors, thus playing a crucial role in constructing effective hedging strategies.

The construction of this state space and the Features DataFrame offers several benefits. It provides the models with a detailed and comprehensive snapshot of current market dynamics, enabling a nuanced understanding that aligns strategies with current financial conditions. The structure of the DataFrame also facilitates easy access and manipulation of the feature inputs,

allowing for targeted testing of various feature combinations. This adaptability ensures that our hedging strategies are both robust and responsive, crucial for navigating the complexities of financial markets and achieving strategic objectives efficiently.

4.3 Reward Function

In our RL framework, the reward function is critical as it provides feedback to the agent by assigning a numerical value to each action taken within a state. This function is designed to guide the agent to maximise cumulative rewards over time, thereby learning optimal behaviour for hedging strategies. The reward function is defined as follows:

$$\frac{(nS_1 \cdot S_1 - nS_0 \cdot S_0) - num_sold_opt \cdot (C_1 - C_0) - [(nS_1 - nS_0) \cdot S_0 \cdot e^{r(T-T_0)}] - [\alpha \cdot S_0 \cdot (|nS_1 - nS_0| + 0.01 \cdot (nS_1 - nS_0)^2)] - \beta \cdot std(p\&l)}{\gamma^{T_0}}$$

The first component is stock profit and loss $(nS_1 \cdot S_1 - nS_0 \cdot S_0)$. This measures the profit or loss from holding a specific quantity of stock shares, directly reflecting the financial results of the agent's stock holdings over a given period.

The second component is option price changes, which is $-\ num_sold_opt \cdot (C_1 - C_0)$. This assesses the impact of changes in option prices on the agent's reward. The negative sign indicates that the agent is shorting the options; therefore, the agent gains if the option prices decrease and loses if they increase. This directly links the volatility of the option market to the agent's performance, highlighting the financial implications of price movements on short positions.

The third component of the reward function is the stock position adjustment cost $[(nS_1 - nS_0) \cdot S_0 \cdot e^{r(T-T_0)}]$. This represents the opportunity cost associated with adjusting the stock position, calculated based on the financing costs of changes in the stock position using the risk-free rate r. This cost acknowledges the financial implications of maintaining or modifying hedge positions over time.

The additional transaction cost component $[\alpha \cdot S_0 \cdot (|nS_1 - nS_0| + 0.01 \cdot (nS_1 - nS_0)^2)]$ captures the explicit costs associated with executing trades, such as commissions and fees. This penalises large or frequent trades to discourage excessive trading that could erode profits through high transaction costs.

Lastly, since we want to reduce the volatility of our portfolio to encourage stable returns, we deduct the reward by $\beta \cdot std(p\&l)$, which is a scaled value of the standard deviation of P&L over the last 30 days.

To prioritise immediate gains that are crucial for effective hedging, the reward function incorporates a discounting mechanism using the Gamma factor, raised to the power of the

current time, γ^{T_0}. This discounting emphasises the importance of near-term rewards over distant future rewards, aligning the agent's actions with the short-term objectives of the hedging strategy. Collectively, these components ensure that the reward function comprehensively addresses both the direct and indirect costs and benefits of trading actions, guiding the agent towards decisions that optimise the overall hedging strategy.

The code implementation for the reward function can be found in Appendix 2.

4.4 Action Space

In our RL framework, the implementation of each agent is designed to determine the optimal action at each timestep based on the current state of the environment. The action taken represents the desired stock position at the subsequent time, t+1, and each action function returns a singular value corresponding to this position.

For the strategy of Delta Hedging, the action dictated by the agent corresponds directly to the delta value of the current state. This method straightforwardly adjusts the stock position to align with the delta of the options, aiming for an ideally hedged portfolio.

For the DQN agent, it employs an epsilon-greedy strategy. This approach involves a transition from random exploration of actions to a more refined strategy that increasingly relies on selecting actions based on the highest Q-values as more episodes are processed.

For the DDPG agent, the action is determined by its Actor network, which computes the optimal action for the current state. Concurrently, the Critic network evaluates the action by estimating the Q-value and plays a crucial role when the Actor updates its weights. This interplay between the Actor and Critic ensures that the actions are not only optimal based on past learning but are also continually refined to better suit the evolving market conditions.

5.0 Training and Testing Procedures

In order to test and train the model, we split the stock data into two parts: training data for the time period between 01/01/2010 to 31/12/2022, and testing data for the time period between 01/01/2023 to the current date, and generated separate options data for each. We created two classes: train_agent, and test_agent, to perform the training and testing over episodes, each utilising mutually exclusive datasets, ensuring that test evaluation will only be on unseen data.

5.1 Training the Models

The train_agent class trains our models over a specified period of 1000 episodes. Each episode is designed to simulate the entire lifecycle of an option, from its inception to maturity. Hence, each episode spans n steps, matching the time to maturity of the option T. This setup allows the agent to simulate hedging the option through its entire lifetime in a

controlled environment, mirroring real-world trading timelines and decision-making scenarios closely.

The training process begins with the environment selecting a random starting point within the training dataset for each episode. During each step, the agent assesses the current state to decide on an action, subsequently receiving a reward and a new state from the environment. This sequential interaction is captured and stored in the agent's memory to facilitate learning.

Each step's outcomes contribute to the update of the agent's total reward, which is adjusted by a discount factor to align with short-term priorities. The portfolio's profit and loss (P&L) are meticulously tracked at each step. This constant updating of the agent's memory and policies ensures a robust learning cycle, enabling the agent to adapt and refine its strategies based on past experiences. The training reward and P&L over episodes for the DQN and DDPG models can be found in Appendix 3.

5.2 Testing the Models

The testing phase employs a similar structure with the test_agent, which also runs over 1000 episodes. However, it utilises an unseen test dataset to evaluate the model's performance under new conditions. Similar to training, each test episode starts at a randomly chosen point within the test dataset. The agent follows a deterministic policy during testing, which means it selects actions based on the learned strategies from the training phase, without random exploration. This phase is critical for assessing the robustness and generalizability of the trained models across different market conditions not previously encountered during the training phase.

5.3 Code Implementation

In the code folder, there are 6 files: Generate Data, Environment, Delta_Baseline, DQN, DDPG and Model_Eval. Generate Data pulls the yfinance data, does further feature engineering, and creates the simulated options data. This ultimately creates the train and test datasets. The Environment file defines the environment class, including the logic for what happens at each step, at the start of each episode, and has functions to calculate the reward as well as profit and loss (PL). Delta_Baseline, DQN, and DDPG files act as the agents and each of them implement their own actor, and train the models based on the implemented environment and the datasets generated. Then, in model evaluation, all the saved models are imported, and tests the models on the test dataset so they can be evaluated against one another.

6.0 Evaluation Metrics

It is important to note that the fundamental goal of hedging is to mitigate risk, prioritising the protection of investments from unfavourable market movements over maximising profits. In evaluating our RL models, it is crucial to employ robust performance metrics that align with these primary objectives of financial risk management.

In our research, we focus on three critical performance metrics: **Mean Profit and Loss (P&L)**, **Standard Deviation (SD) of P&L**, and the **1st Percentile P&L**. These metrics are widely recognized in financial literature and are commonly used by options traders to evaluate hedging strategies.

Mean P&L indicates the average profit or loss over a specific period, providing a baseline for the strategy's overall profitability. While profitability is important, it is considered secondary to the strategy's risk management capabilities in a hedging context.

Standard Deviation of P&L measures the variability of profits and losses, quantifying the consistency of the strategy's performance. A lower standard deviation is crucial for hedging strategies as it indicates more stable returns, thus delivering predictable protection against market fluctuations.

1st Percentile P&L assesses potential losses in the worst 1% of scenarios, highlighting the strategy's robustness under severe market conditions. This measure is critical for understanding the strategy's performance during extreme market events, providing insights into its tail risk management.

7.0 Experimental Results and Analysis

We first run the models without changing any parameters ($\alpha = 1 \times 10^{-3}$, T = 30), while using the full state space that we created through feature engineering. This allows us to first do a baseline comparison between the different model's performance on test data. The results below show that both DQN and DDPG performed better than the baseline delta model, having a lower standard deviation as well as a better 1 Percentile P&L, showing that the models were able to obtain more stable returns while having a smaller maximum loss. Further experimentation will be done to see whether the models still continue outperforming the baseline model in different simulated market conditions.

Agent	Mean P&L	Std. Dev. P&L	1 Percentile P&L
Delta	24.21	463.41	-1001.31
DQN	-27.71	448.99	-982.10
DDPG	67.34	433.15	-861.60

Table 1: Statistics of P&L after 1000 episodes with $\alpha = 1 \times 10^{-3}$

7.1 Cases of Varying Non-zero Transaction Costs

We investigate the impact of an increase in transaction costs by adjusting the parameter α for multiple purposes. Firstly, it realistically reflects the variations in transaction costs that can occur across different markets or with different types of options. Secondly, by elevating transaction costs, we can assess how well each model performs in environments that penalise frequent trading. This manipulation of α effectively simulates conditions that favour less frequent hedging, shifting from dynamic to more static hedging approaches.

This methodological adjustment allows us to identify which model performs best under the pressure of minimised trading activities, offering valuable insights into each model's ability to adapt to scenarios where static hedging is preferable due to high transaction costs. Such a nuanced approach not only mirrors the complexities of real-world trading but also rigorously tests the adaptability and resilience of our hedging strategies in environments where trading activities are financially constrained.

Initially, with α set to 1×10^{-3}, the DDPG agent demonstrated the best performance across all metrics, suggesting that moderate transaction costs do not drastically impair the ability to conduct effective dynamic hedging. Specifically, DDPG managed to maintain a robust performance in terms of Mean P&L, Standard Deviation of P&L, and 1st Percentile P&L, indicating its efficiency under the cost conditions of $\alpha = 1 \times 10^{-3}$ (Table 1). This outcome highlights that DDPG can balance cost-efficiency with strategic hedging effectiveness.

As the transaction cost parameter α was increased to 4×10^{-3} and then to 8×10^{-3}, a clear trend emerged where both DQN and DDPG began to outperform the Delta agent significantly. This observation is particularly notable in the context of Standard Deviation of P&L, which decreased for DQN and DDPG, suggesting a more stable performance despite higher costs (Tables 2 and 3). The DQN agent, in particular, showed a fair ability to adapt to higher transaction costs, performing better than DDPG. This adaptation indicates DQN's potential as a more cost-efficient model under conditions of high transaction costs, suitable for settings that mimic static hedging.

With the highest increase in transaction costs $\alpha = 8 \times 10^{-3}$, both DQN and DDPG agents showed an ability to adapt by reducing the variability in their P&L outcomes. DQN not only maintained a lower Standard Deviation of P&L but also achieved a significantly better Mean P&L and 1st Percentile P&L compared to DDPG and Delta. This performance underscores DQN's capability to effectively adjust its trading frequency and strategy in response to penalising transaction cost environments.

Agent	Mean P&L	Std. Dev. P&L	1 Percentile P&L
Delta	-125.11	428.58	-1013.02
DQN	-7.85	422.39	-1001.35
DDPG	-152.31	435.30	-1159.43

Table 2: Statistics of P&L after 1000 episodes with $\alpha = 4 \times 10^{-3}$

Agent	Mean P&L	Std. Dev. P&L	1 Percentile P&L
Delta	-376.22	413.58	-1246.89
DQN	-31.10	406.43	-871.82
DDPG	-374.76	406.08	-1225.10

Table 3: Statistics of P&L after 1000 episodes with $\alpha = 8 \times 10^{-3}$

7.2 Cases of Hedging Options with Varying Maturities

Reverting the transaction cost parameter α back to 0.001, we continued our exploration into the impact of varying option maturities, denoted as T, on the performance of our models. This analysis is critical as it helps to understand how well each model can adapt its hedging strategies over different time horizons, reflecting their applicability to real-world trading scenarios where option maturities can significantly influence strategic decisions.

For a maturity of 30 days (T=30), the DDPG model exhibited superior performance, enjoying the lowest standard deviation of P&L, which demonstrates its effectiveness in managing short-term hedges. This result highlights DDPG's robust capability in quickly adjusting to market conditions over shorter periods.

As the maturity was extended to 60 days (T=60), DDPG continued to show commendable performance by maintaining the lowest standard deviation among the models. This indicates that DDPG offers consistent hedging performance over medium-term horizons, minimising risk while managing returns effectively. Despite DDPG's lower variability in returns, the Delta strategy reported a higher mean P&L, which could be attributed to its higher risk as indicated by its greater standard deviation.

However, when the maturity was further increased to 90 and 120 days (T=90 and T=120), the performance dynamics shifted. In these scenarios, the Delta strategy outperformed both DDPG and DQN models. This shift suggests that while the RL models—DDPG and DQN—are effective for medium-term maturities, they struggle to adapt to the longer-term market conditions and increased uncertainty that come with extended maturities. The superior performance of the Delta strategy in these longer maturities implies that traditional delta hedging may still hold an edge over RL-based strategies when facing extended durations and higher market unpredictability.

These findings indicate that while RL models such as DDPG and DQN demonstrate robust capabilities in shorter and medium-term hedging scenarios, the models would benefit from enhancements to better navigate the complexities and uncertainties of longer-term financial markets. The variation in performance across different maturities highlights the need for more sophisticated modelling techniques that can adapt to the nuanced dynamics of extended time horizons and fluctuating market conditions. This adaptation will ensure that the hedging strategies remain effective and relevant to the specific requirements dictated by the lifecycle of the options.

Agent	Mean P&L	Std. Dev. P&L	1 Percentile P&L
Delta	46.18	404.38	-826.50
DQN	-23.86	401.02	-828.76
DDPG	-199.26	389.34	-1102.03

Table 4: Statistics of P&L after 1000 episodes with T = 60

Agent	Mean P&L	Std. Dev. P&L	1 Percentile P&L
Delta	19.93	387.80	-783.27
DQN	-17.70	407.86	-944.29
DDPG	-200.51	402.87	-1140.00

Table 5: Statistics of P&L after 1000 episodes with T = 90

Agent	Mean P&L	Std. Dev. P&L	1 Percentile P&L
Delta	0.20	403.91	-783.27
DQN	-41.97	406.43	-871.82
DDPG	-207.13	405.04	-1075.32

Table 6: Statistics of P&L after 1000 episodes with T = 120

7.3 Modifying the State Space

In the sections 7.3.1 and 7.3.2 below, we delve into the effects of selectively removing key components such as delta and call option prices from the state space. This exploration is designed to assess how our RL models perform when they are less reliant on direct pricing signals, such as those derived under the Black-Scholes-Merton (BSM) model, and instead draw on a wider range of market and technical indicators. Such an approach not only challenges the models to adapt to less explicit information—a crucial requirement in real-world scenarios where data may be incomplete or distorted by market anomalies—but also tests their robustness across diverse informational environments.

Including components like the option price and delta in the state space typically offers a closed-loop feedback mechanism that simplifies decision-making by providing "the right answers" for next actions as per theoretical models. These components encapsulate essential information for adjusting hedges at each step based on the BSM model's assumptions. However, real-world financial markets often deviate from these theoretical assumptions, such as constant volatility or log-normal stock price distributions, thereby challenging the practical utility of these models. This discrepancy highlights the potential risk of relying solely on option price and delta for hedging decisions, especially under atypical market conditions or when closed-form solutions are unavailable.

7.3.1 Analysis of Removing Delta from State Space

We explored the effects of excluding the delta component from the state space to evaluate the dependency of our RL models on this specific metric. Delta plays a fundamental role in dynamic hedging as it directly informs decisions on the number of shares to buy or sell to maintain a neutral market position. Its removal from the state space strips the model of a direct indicator of exposure to underlying asset price movements, crucial for accurately adjusting hedging positions. Consequently, the absence of delta challenges the model to devise hedging strategies without clear insights into market sensitivities, potentially leading to heightened risk and variability in performance.

Initially, with delta included, the DDPG model outperformed both RL models across all key performance metrics (Table 1). However, the removal of delta presented a significant shift in model performance (Table 7).

Agent	Mean P&L	Std. Dev. P&L	1 Percentile P&L
Delta*	24.21	463.41	-1001.31
DQN	-18.44	456.55	-938.28
DDPG	-207.45	453.24	-1140.86

Table 7: Statistics of P&L after 1000 episodes with delta not in state space
* Delta agent results taken from Table 1 (Delta Agent will not run without delta in state space)

Upon removing delta from the state space, we observed an increase in the standard deviation of P&L for all models, indicating a rise in volatility and risk in the hedging decisions made by each agent. Despite this increase, DDPG maintained a lower standard deviation compared to the other models, suggesting that while DDPG is affected by the absence of delta, it remains relatively more stable than its counterparts.

These findings suggest a strong reliance on the delta component by the RL models to make effective hedging decisions. Without access to delta, the models struggled to adequately leverage other fundamental variables such as the stock price S_t, strike price K, holdings N_t,

and additional relevant market data. This limitation highlights a gap in the models' ability to infer optimal hedging strategies using solely fundamental market features, indicating a potential area for model improvement.

To enhance the robustness and adaptability of these models, further development is necessary. This could involve integrating more sophisticated machine learning techniques or enhancing feature engineering to better capture the dynamics of the market without heavily relying on delta. Such improvements would aim to equip the models with the capability to construct effective hedging strategies using a broader set of market data inputs.

7.3.2 Analysis of Removing Option Price from State Space

After reinstating the delta component into the state space, our subsequent analysis focused on evaluating the effects of removing the option price. When the option price was excluded, the Delta strategy consistently outperformed both RL models (DQN and DDPG) across all three performance metrics (Table 8). Notably, while DQN and DDPG did not surpass the Delta strategy, both models exhibited an improvement in the standard deviation of P&L, indicating a potentially more stable strategy when option prices are not directly used in their decision-making process.

Agent	Mean P&L	Std. Dev. P&L	1 Percentile P&L
Delta	24.21	463.41	-1001.31
DQN	10.18	432.75	-1012.02
DDPG	-203.60	425.54	-1164.80

Table 8: Statistics of P&L after 1000 episodes with option price not in state space

This stability improvement can be attributed to the elimination of circularity in the models' learning mechanisms. When we simulated the options, the call option price is derived from other variables already present in the state space such as stock price and volatility. This setup can lead to a feedback loop where models might overfit to the nuances of option price calculations. By removing the option price, models are compelled to rely more on fundamental market variables like stock price, volatility, and other financial indicators. This shift likely fosters a more robust learning environment, reducing dependency on the immediate fluctuations in option prices which can be highly volatile and reactive to market sentiments rather than underlying financial principles.

Notably, this contrasts with the removal of delta, which did not enhance model performance. The elimination of the call option price from the state space did contribute positively for a few possible reasons. While the call option price is a derivative measure under the Black-Scholes framework, it does not provide actionable insights comparable to delta for direct hedging adjustments. Its removal led to a reduction in the model's tendency to 'echo' its training data, decreasing redundancy and potential feedback loops within the learning process. This crucial adjustment contributed to the observed decrease in standard deviation,

indicating a shift towards a hedging strategy that relies more broadly on interpreting market data. Such a broader reliance may enhance the model's capacity to generalise from available information, suggesting a more robust and adaptable approach to hedging.

These insights highlight the nuanced impact of adjusting the state space. Removing delta did not improve performance, whereas eliminating the call option price reduced dependency on redundant data, underscoring the importance of tailored strategies that consider the specific role of each component in financial modelling.

7.3.3 Analysis of Removing Other Components in Features DataFrame

RL models are distinguished by their dynamic capability to interact with their environment and derive optimal strategies through a process of trial and error. Traditionally, these models adapt their decision-making processes based on the feedback they receive, which often reduces the necessity for manual feature selection. This adaptive trait allows RL models to autonomously discern and prioritise the most pertinent features from intricate datasets.

Despite the inherent adaptability of RL models, our research seeks to probe deeper into the strategic importance of feature selection within these systems. We aim to discern the impact of omitting certain features, considered advantageous in aiding traders within financial sectors, on the models' performance in realistic trading scenarios. Our approach involves controlled experiments to systematically alter the state space, exploring how RL models cope with diminished information and their efficacy under these conditions.

This approach allows us to challenge the models beyond standard operational settings, testing their robustness and adaptability under conditions of limited data. This kind of rigorous testing is crucial for deploying RL models in financial markets, where unpredictability and incomplete information are common. Through these experiments, we explore the true resilience and flexibility of RL algorithms, providing valuable insights into their ability to generalise from available data.

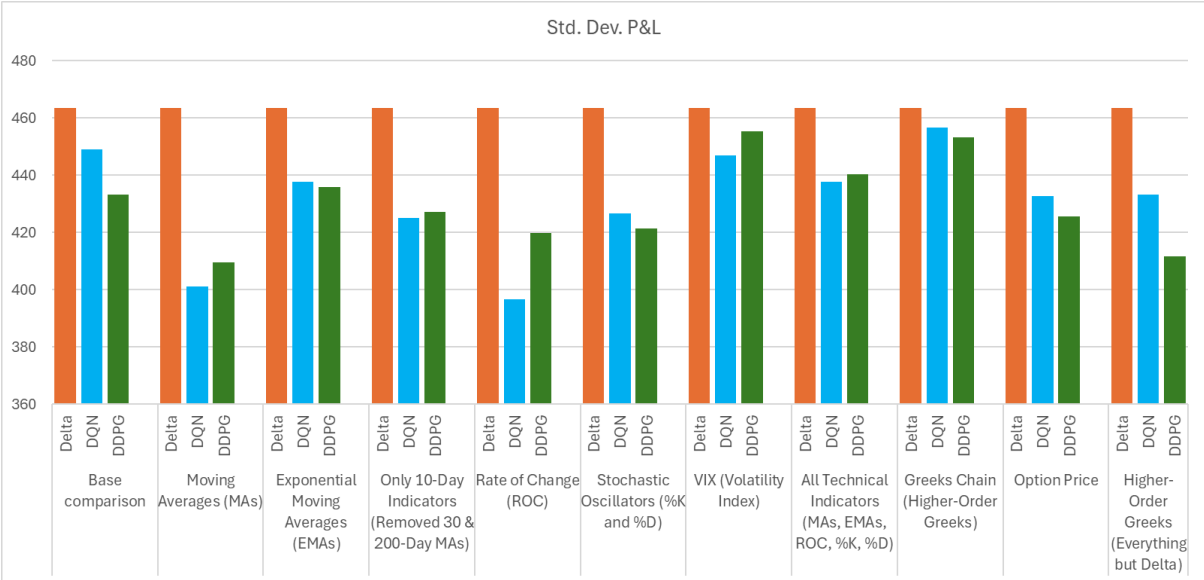


Figure 5: Column Chart comparing Standard Deviation of P&L across different state spaces for each model

In these experiments, we methodically removed individual features from the DataFrame, focusing solely on the standard deviation of P&L as the evaluative metric. Each feature listed in the column chart in Figure 5 corresponds to a feature that was removed from the state space for the purpose of the experiment. An increase in standard deviation of P&L following the removal of a feature suggests its beneficial role in enhancing the model's decision-making capabilities. The choice to concentrate on standard deviation of P&L aligns with the objective in hedging strategies to minimise volatility, crucial for effective risk management. It also reflects the practical goal of surpassing performance benchmarks, such as Delta Hedging, even when achieving perfect stability (zero SD of P&L) remains unfeasible due to inherent market volatilities. For the detailed evaluation results for each feature set, refer to Appendix 4.

Our experimental goal was to avoid burdening the model with superfluous features that might introduce noise or unwarranted complexity. Thus, the focus was placed on high-impact features that demonstrably reduce standard deviation of P&L, while excluding redundant indicators that do not contribute to risk mitigation. In these experiments, the removal of features such as EMA, VIX, and delta led to either the same or increased volatility, highlighting their essential role in stabilising the model and managing risk effectively.

Therefore, an attempt to reduce the Feature DataFrame to include only EMA, VIX and delta to optimise the hedging strategies effectively and avoid incorporating redundant or noise-inducing variables, was made. However, after running the model, the SD of PL increased compared to the base scenario rather than decreasing (Figure 6). This shows that features that do not show significant individual benefits can in fact significantly enhance models' performances when they interact with other features. It also suggests a complex relationship where these features contribute nuanced, valuable information that aids in

accurate predictions. Removing features seemed prudent initially but led to a loss of crucial signals, making our model more sensitive to fluctuations in the remaining data.

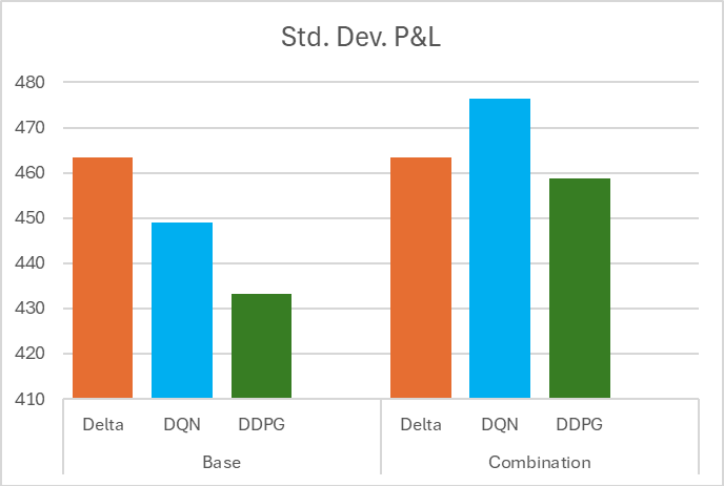


Figure 6: Column Chart comparing Standard Deviation of P&L when state space only contains EMA, VIX, and delta

Some potential reasons for the results may be due to the interplay between the features where they work together to enable the model to ultimately make better decisions. It may also be because of the loss of information with the removal of features and with less features, the models may also then be more susceptible to the noise in the market, not allowing the models to have a holistic understanding of the market conditions.

This experiment underscores the importance of holistic feature evaluation, where the collective impact of features is considered rather than judging on an individual basis. Hence, it may be wise to explore iterative testing of different combinations instead. Furthermore, feature selection will not be needed if we utilise a more complex RL model since the learning process will automatically select and utilise the relevant features needed to help the model make the best decisions.

8.0 Conclusion and Future Directions

From the study, we can conclude that DDPG is the better model for most cases. The DDPG agent has a smaller standard deviation of profit and loss than the DQN agent in most of the numerical experiments presented in this report.

Going forward, it would be advantageous to conduct further experiments to explore the implications of integrating or excluding different types of market data in the state space, as mentioned above. Additionally, integrating complex market dynamics such as multi-asset interactions could significantly bolster the models' resilience by enabling them to adapt to unpredictable conditions and better generalise across varied market environments.

Furthermore, we could try to increase the depth of our DQN and DDPG models, as we are currently using at most a 3 layer neural network architecture. With more time and computational resources, we could have implemented a deeper architecture which may allow the model to learn more of the complexities in the given dataset, given that we also performed comprehensive feature engineering that may need a more complex model to capture.

We think that it would also be valuable to compare against a wider range of reinforcement learning models, including Distributionally Robust Reinforcement Learning (DRRL) and advanced algorithms like Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC). This would provide deeper insights into optimising portfolio performance under diverse conditions. These models might also offer improved robustness against the inherent uncertainties of financial data.

Lastly, further research could investigate whether stacking agents with statistically diverse policies can effectively reduce the variance of rewards and P&L, contributing to a more stable and reliable hedging strategy. Model stacking, which involves averaging policies from different models such as DQN and DDPG, might prove advantageous if applied in real-life trading, potentially enhancing the stability of the models under market fluctuations.

Appendices

Appendix 1

Figure 7

Black-Scholes model implemented

```python
def _black_scholes_greeks(self, S, K, T, r, sigma):
    """
    Calculate the Black-Scholes call option price and Greeks.
    """
    if T <= 0 or sigma <= 0 or S <= 0 or K <= 0:
        return 0, 0, 0, 0, 0, 0

    d1 = (np.log(S / K) + (r + 0.5 * sigma ** 2) * T) / (sigma * np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)

    call_price = S * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2)
    delta = norm.cdf(d1)
    gamma = norm.pdf(d1) / (S * sigma * np.sqrt(T))
    theta = (-S * norm.pdf(d1) * sigma / (2 * np.sqrt(T))
             - r * K * np.exp(-r * T) * norm.cdf(d2))
    vega = S * norm.pdf(d1) * np.sqrt(T) / 100  # Per 1% change in volatility
    rho = K * T * np.exp(-r * T) * norm.cdf(d2) / 100  # Per 1% change in rate

    return call_price, delta, gamma, theta, vega, rho
```

Appendix 2

Figure 8

Reward Function

```python
def _calculate_reward(
    self, T0=None, T1=None, S0=None, S1=None, delta0=None, delta1=None,
    nS0=None, nS1=None, vol=None, action=None, C0=None, C1=None, terminal=False
):
    if terminal:
        # Terminal reward calculation
        initial_option_price = self.option_data['OptionPrice'].iloc[self.initial_idx_time]
        r = self.num_sold_opt * (np.exp(self._get_risk_free_rate() * self.T) - 1) * initial_option_price
        r = r / np.power(self.gamma, self.n_steps - self.T)
        return r

    # Check for NaN values in inputs
    if any(np.isnan([S0, S1, delta0, delta1, nS0, nS1, vol, action, C0, C1])):
        print(f"NaN detected in reward calculation inputs: S0={S0}, S1={S1}, delta0={delta0}, delta1={delta1}, nS0={nS0}, nS1={nS1}, vol={vol
        return 0.0  # Or handle as per your logic

    r = self._get_risk_free_rate()

    reward = nS1 * S1 - nS0 * S0
    reward -= self.num_sold_opt * (C1 - C0)
    reward -= (nS1 - nS0) * S0 * np.exp(r * (self.T - T0))

    if self.alpha > 0:
        cost = self.__get_cost(S=S0, chg_nS=(nS1 - nS0))
        reward -= cost * np.exp(r * (self.T - T0))

    if self.kappa > 0:
        var = vol * S0 * (nS1 - delta0)
        var = var**2 * self.dt
        reward -= self.kappa * var / 2

    # Incorporate standard deviation of the last 30-day PnL
    if len(self.pnl_history) > 1:
        pnl_std = np.std(self.pnl_history)
        reward -= self.beta * pnl_std  # Penalize high standard deviation
    else:
        pnl_std = 0.0  # No penalty if insufficient history

    reward = reward / np.power(self.gamma, T0)

    return reward
```

Appendix 3

Graph of reward and SD of PL

Figure 9

DQN's results

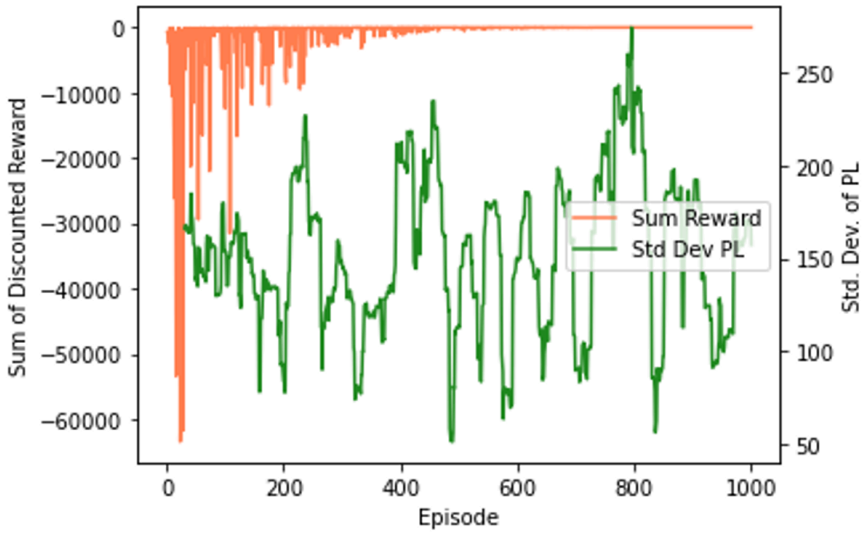
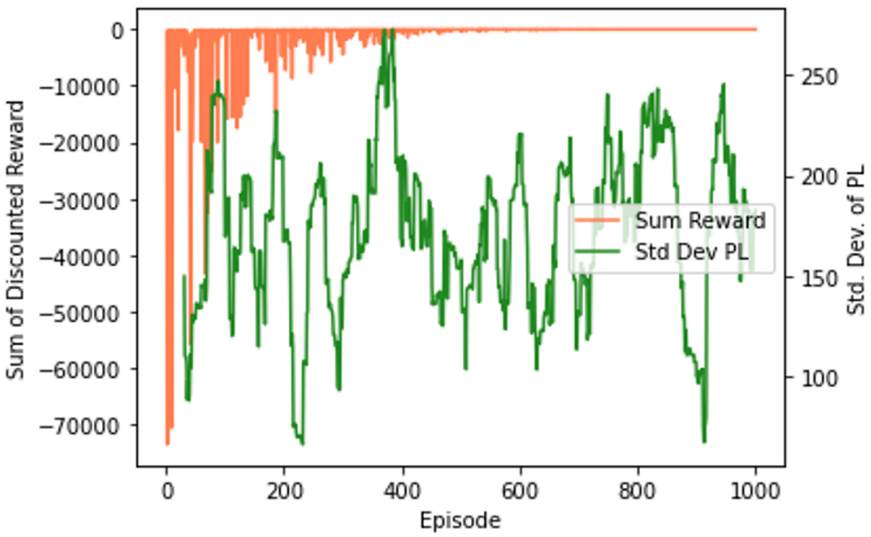


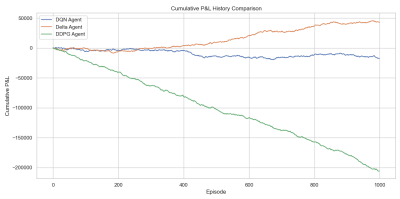
Figure 10

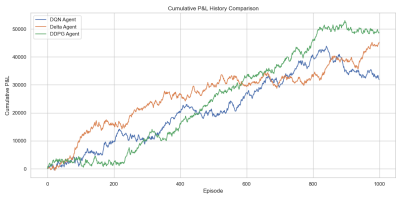
DDPG's results

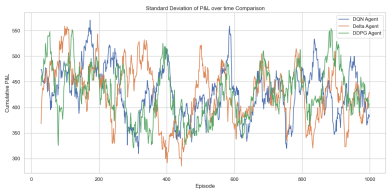
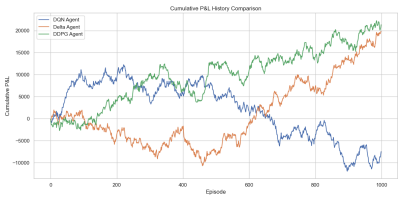


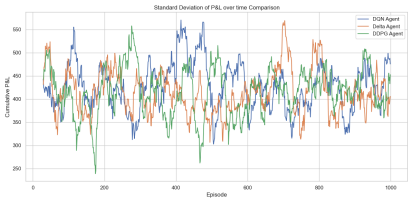
Appendix 4

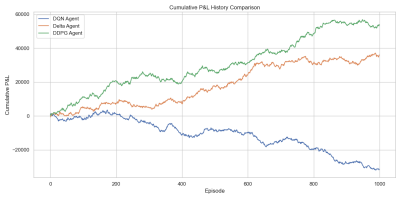
Evaluation results for each experiment

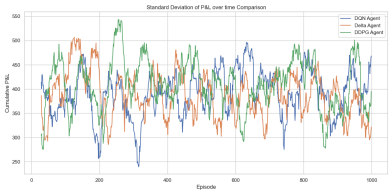
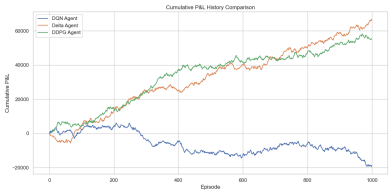
<table>
<tr><th>Change/ Simulation:</th><th>P&L & SD graphs</th><th>Numerical results:</th></tr>
<tr><td>MAs</td><td> 

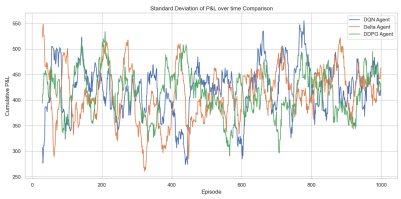
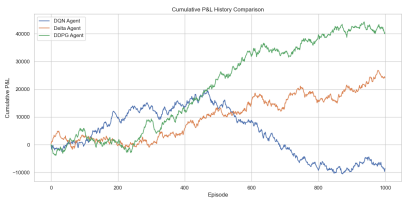
</td><td>
<table>
<tr><th>Agent</th><th>Std. Dev. P&L</th><th>1 Percentile P&L</th><th>Mean P&L</th></tr>
<tr><td>Delta</td><td>24.21</td><td>463.41</td><td>-1001.31</td></tr>
<tr><td>DQN</td><td>-17.76</td><td>401.11</td><td>-881.58</td></tr>
<tr><td>DDPG</td><td>-205.83</td><td>409.52</td><td>-1142.37</td></tr>
</table>
</td></tr>
<tr><td>EMAs</td><td> 

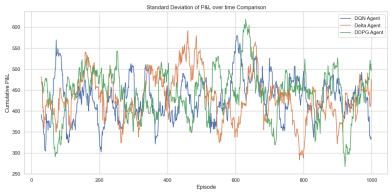
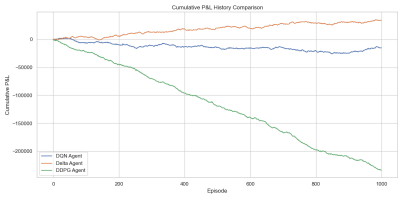

</td><td>
<table>
<tr><th>Agent</th><th>Std. Dev. P&L</th><th>1 Percentile P&L</th><th>Mean P&L</th></tr>
<tr><td>Delta</td><td>24.21</td><td>463.41</td><td>-1001.31</td></tr>
<tr><td>DQN</td><td>31.87</td><td>437.66</td><td>-857.21</td></tr>
<tr><td>DDPG</td><td>48.66</td><td>435.83</td><td>-817.00</td></tr>
</table>
</td></tr>
<tr><td>30 and 200 so leave only the 10 days indicators</td><td> 

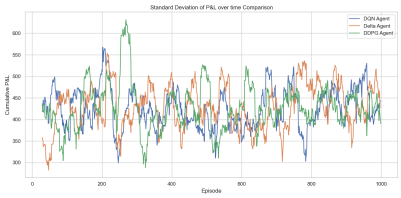
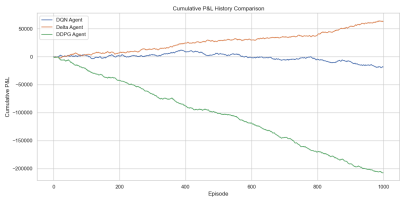

</td><td>
<table>
<tr><th>Agent</th><th>Std. Dev. P&L</th><th>1 Percentile P&L</th><th>Mean P&L</th></tr>
<tr><td>Delta</td><td>24.21</td><td>463.41</td><td>-1001.31</td></tr>
<tr><td>DQN</td><td>-7.47</td><td>424.96</td><td>-990.83</td></tr>
<tr><td>DDPG</td><td>21.33</td><td>427.06</td><td>-1001.31</td></tr>
</table>
</td></tr>
</table>

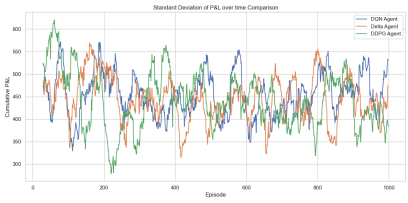
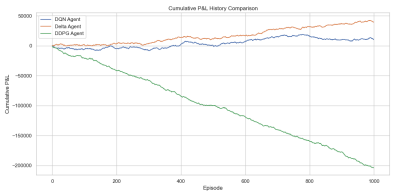
<table>
<tr><th>Change/ Simulation:</th><th>P&L & SD graphs</th><th>Numerical results:</th></tr>
<tr><td>ROC</td><td> 

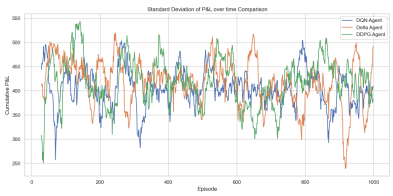
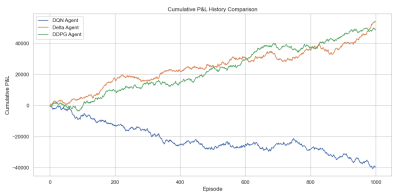

</td><td>
<table>
<tr><th>Agent</th><th>Std. Dev. P&L</th><th>1 Percentile P&L</th><th>Mean P&L</th></tr>
<tr><td>Delta</td><td>24.21</td><td>463.41</td><td>-1001.31</td></tr>
<tr><td>DQN</td><td>-31.35</td><td>396.55</td><td>-823.37</td></tr>
<tr><td>DDPG</td><td>53.97</td><td>419.77</td><td>-797.81</td></tr>
</table>
</td></tr>
<tr><td>%k and %d</td><td> 

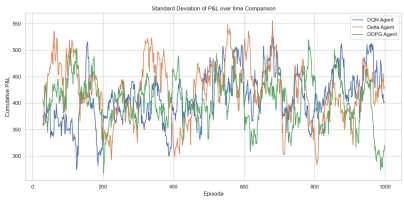

</td><td>
<table>
<tr><th>Agent</th><th>Std. Dev. P&L</th><th>1 Percentile P&L</th><th>Mean P&L</th></tr>
<tr><td>Delta</td><td>24.21</td><td>463.41</td><td>-1001.31</td></tr>
<tr><td>DQN</td><td>-19.49</td><td>426.64</td><td>-1008.74</td></tr>
<tr><td>DDPG</td><td>55.53</td><td>421.45</td><td>-944.38</td></tr>
</table>
</td></tr>
<tr><td>VIX</td><td> 

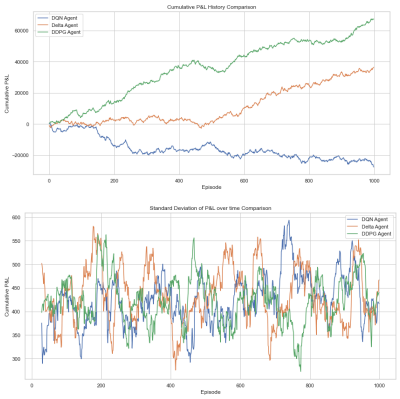
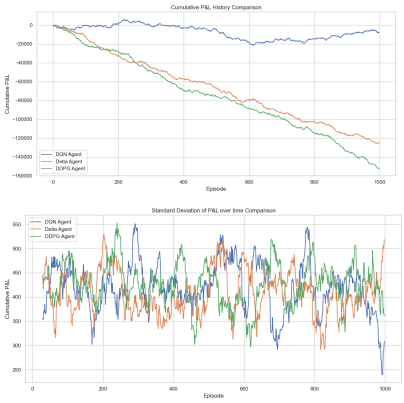
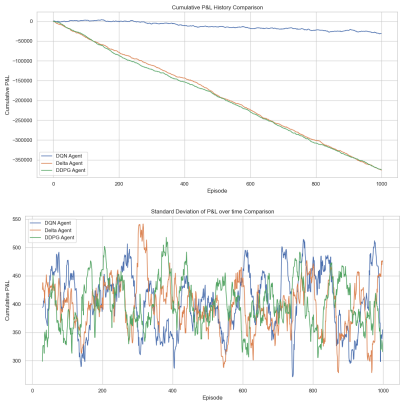
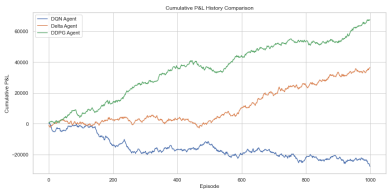

</td><td>
<table>
<tr><th>Agent</th><th>Std. Dev. P&L</th><th>1 Percentile P&L</th><th>Mean P&L</th></tr>
<tr><td>Delta</td><td>24.21</td><td>463.41</td><td>-1001.31</td></tr>
<tr><td>DQN</td><td>-8.48</td><td>446.97</td><td>-1014.86</td></tr>
<tr><td>DDPG</td><td>40.28</td><td>455.32</td><td>-960.70</td></tr>
</table>
</td></tr>
<tr><td>All MAs, EMA, ROC, %K, %D</td><td> 
</td><td>
<table>
<tr><th>Agent</th><th>Std. Dev. P&L</th><th>1 Percentile P&L</th><th>Mean P&L</th></tr>
<tr><td>Delta</td><td>24.21</td><td>463.41</td><td>-1001.31</td></tr>
</table>
</td></tr>
</table>

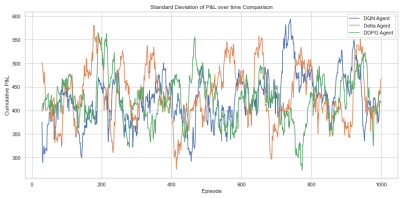
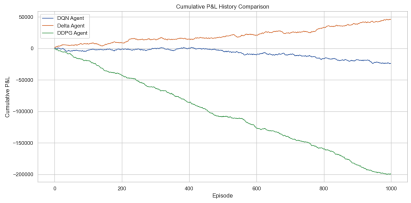
<table>
<tr><th>Change/ Simulation:</th><th>P&L & SD graphs</th><th>Numerical results:</th></tr>
<tr><td></td><td> 
</td><td>
<table>
<tr><td>DQN</td><td>-14.97</td><td>437.82</td><td>-896.57</td></tr>
<tr><td>DDPG</td><td>-233.92</td><td>440.36</td><td>-1067.40</td></tr>
</table>
</td></tr>
<tr><td>Greeks chain</td><td> 

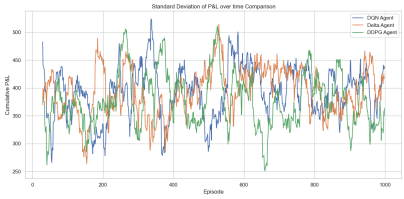
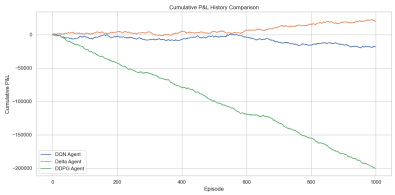

</td><td>
<table>
<tr><th>Agent</th><th>Std. Dev. P&L</th><th>1 Percentile P&L</th><th>Mean P&L</th></tr>
<tr><td>Delta</td><td>24.21</td><td>463.41</td><td>-1001.31</td></tr>
<tr><td>DQN</td><td>-18.44</td><td>456.55</td><td>-938.28</td></tr>
<tr><td>DDPG</td><td>-207.45</td><td>453.24</td><td>-1140.86</td></tr>
</table>
</td></tr>
<tr><td>Option price</td><td> 

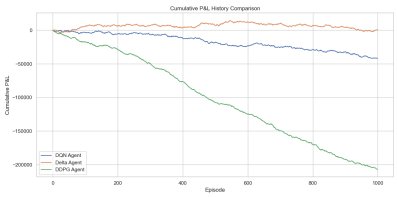

</td><td>
<table>
<tr><th>Agent</th><th>Std. Dev. P&L</th><th>1 Percentile P&L</th><th>Mean P&L</th></tr>
<tr><td>Delta</td><td>24.21</td><td>463.41</td><td>-1001.31</td></tr>
<tr><td>DQN</td><td>10.18</td><td>432.75</td><td>-1012.02</td></tr>
<tr><td>DDPG</td><td>-203.60</td><td>425.54</td><td>-1164.80</td></tr>
</table>
</td></tr>
<tr><td>Higher order greeks (everything but delta)</td><td> 

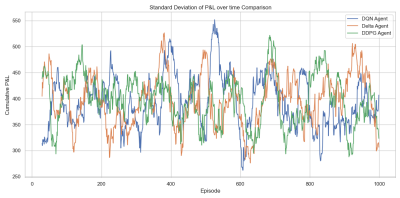
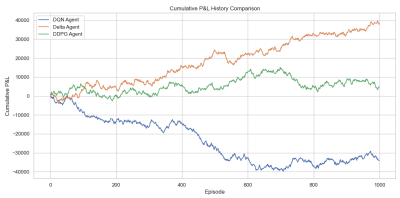

</td><td>
<table>
<tr><th>Agent</th><th>Std. Dev. P&L</th><th>1 Percentile P&L</th><th>Mean P&L</th></tr>
<tr><td>Delta</td><td>24.21</td><td>463.41</td><td>-1001.31</td></tr>
<tr><td>DQN</td><td>-39.61</td><td>433.14</td><td>-956.51</td></tr>
<tr><td>DDPG</td><td>48.63</td><td>411.64</td><td>-895.55</td></tr>
</table>
</td></tr>
</table>

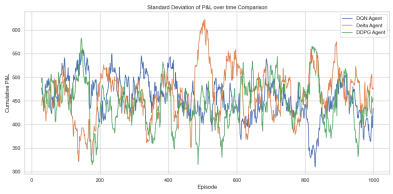
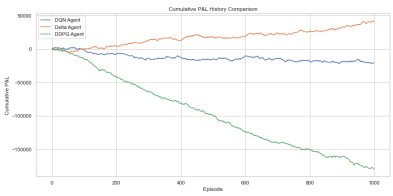
<table>
<tr><th>Change/ Simulation:</th><th>P&L & SD graphs</th><th>Numerical results:</th></tr>
<tr><td>alpha= 0.001 (original)</td><td> 
</td><td>
<table>
<tr><th>Agent</th><th>Std. Dev. P&L</th><th>1 Percentile P&L</th><th>Mean P&L</th></tr>
<tr><td>Delta</td><td>24.21</td><td>463.41</td><td>-1001.31</td></tr>
<tr><td>DQN</td><td>-27.71</td><td>448.99</td><td>-982.10</td></tr>
<tr><td>DDPG</td><td>67.34</td><td>433.15</td><td>-861.60</td></tr>
</table>
</td></tr>
<tr><td>alpha= 0.004</td><td> 
</td><td>
<table>
<tr><th>Agent</th><th>Std. Dev. P&L</th><th>1 Percentile P&L</th><th>Mean P&L</th></tr>
<tr><td>Delta</td><td>-125.11</td><td>428.58</td><td>-1013.02</td></tr>
<tr><td>DQN</td><td>-7.85</td><td>422.39</td><td>-1001.35</td></tr>
<tr><td>DDPG</td><td>-152.31</td><td>435.30</td><td>-1159.43</td></tr>
</table>
</td></tr>
<tr><td>alpha= 0.008</td><td> 
</td><td>
<table>
<tr><th>Agent</th><th>Std. Dev. P&L</th><th>1 Percentile P&L</th><th>Mean P&L</th></tr>
<tr><td>Delta</td><td>-376.22</td><td>413.58</td><td>-1246.89</td></tr>
<tr><td>DQN</td><td>-31.10</td><td>406.43</td><td>-871.82</td></tr>
<tr><td>DDPG</td><td>-374.76</td><td>406.08</td><td>-1225.10</td></tr>
</table>
</td></tr>
<tr><td>T=30 (same as alpha=0.001;</td><td> 
</td><td>
<table>
<tr><th>Agent</th><th>Std. Dev. P&L</th><th>1 Percentile P&L</th><th>Mean P&L</th></tr>
</table>
</td></tr>
</table>

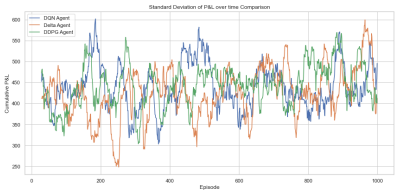
<table>
<tr><th>Change/ Simulation:</th><th>P&L & SD graphs</th><th>Numerical results:</th></tr>
<tr><td>original)</td><td> 
</td><td>
<table>
<tr><td>Delta</td><td>24.21</td><td>463.41</td><td>-1001.31</td></tr>
<tr><td>DQN</td><td>-27.71</td><td>448.99</td><td>-982.10</td></tr>
<tr><td>DDPG</td><td>67.34</td><td>433.15</td><td>-861.60</td></tr>
</table>
</td></tr>
<tr><td>T=60, num step = 60</td><td> 


</td><td>
<table>
<tr><td>Agent</td><td>Std. Dev. P&L</td><td>1 Percentile P&L</td><td>Mean P&L</td></tr>
<tr><td>Delta</td><td>46.18</td><td>404.38</td><td>-826.50</td></tr>
<tr><td>DQN</td><td>-23.86</td><td>401.02</td><td>-828.76</td></tr>
<tr><td>DDPG</td><td>-199.26</td><td>389.34</td><td>-1102.03</td></tr>
</table>
</td></tr>
<tr><td>T=90, num step = 90</td><td> 

</td><td>
<table>
<tr><td>Agent</td><td>Std. Dev. P&L</td><td>1 Percentile P&L</td><td>Mean P&L</td></tr>
<tr><td>Delta</td><td>19.93</td><td>387.80</td><td>-783.27</td></tr>
<tr><td>DQN</td><td>-17.70</td><td>407.86</td><td>-944.29</td></tr>
<tr><td>DDPG</td><td>-200.51</td><td>402.87</td><td>-1140.00</td></tr>
</table>
</td></tr>
<tr><td>T=120, num step = 120</td><td> 
</td><td>
<table>
<tr><td>Agent</td><td>Std. Dev. P&L</td><td>1 Percentile P&L</td><td>Mean P&L</td></tr>
<tr><td>Delta</td><td>0.20</td><td>403.91</td><td>-783.27</td></tr>
<tr><td>DQN</td><td>-41.97</td><td>403.43</td><td>-891.80</td></tr>
<tr><td>DDPG</td><td>-207.13</td><td>405.04</td><td>-1075.32</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>Change/ Simulation:</td><td>P&L & SD graphs</td><td>Numerical results:</td></tr>
<tr><td></td><td> 
</td><td></td></tr>
<tr><td>Without MAs, ROCs, %K %Ds, higher greeks, call option price</td><td> 


</td><td><table>
<tr><th>Agent</th><th>Std. Dev. P&L</th><th>1 Percentile P&L</th><th>Mean P&L</th></tr>
<tr><td>Delta</td><td>24.21</td><td>463.41</td><td>-1001.31</td></tr>
<tr><td>DQN</td><td>-34.04</td><td>475.42</td><td>-993.75</td></tr>
<tr><td>DDPG</td><td>4.73</td><td>457.62</td><td>-932.08</td></tr>
</table></td></tr>
<tr><td>Only keeping EMA, VIX and delta</td><td> 


</td><td><table>
<tr><th>Agent</th><th>Std. Dev. P&L</th><th>1 Percentile P&L</th><th>Mean P&L</th></tr>
<tr><td>Delta</td><td>24.21</td><td>463.41</td><td>-1001.31</td></tr>
<tr><td>DQN</td><td>-21.02</td><td>476.42</td><td>-1022.96</td></tr>
<tr><td>DDPG</td><td>-179.53</td><td>458.74</td><td>-1177.96</td></tr>
</table></td></tr>
</table>