Dissertation

Arandeep Singh Khaira
Department of Computer Science
University of Bath
Bath, BA2 7AY
aak263@bath.ac.uk

September 7, 2025

Contents

1 Abstract

This is the abstract where I'll give an overview. For example this is what I sought to solve and this is what happened etc.

2 Literature Review

2.1 Introduction

Option hedging is traditionally grounded in mathematical finance models and optimal control theory. Classic results like the Black-Scholes-Merton framework showed that a continuously adjusted portfolio could in theory perfectly replicate a European option's payoff, eliminating risk (Black and Scholes 1973; Merton 1973). In practice, however, markets exhibit regime shifts, periods of calm suddenly followed by volatility "storms". Such shifts, alongside factors such as price jumps, transaction costs, and discrete trading leave real-world hedgers with a residual risk (hedging shortfall), thus violating the assumptions of various mathematical models. Reinforcement learning (RL) has recently emerged as a powerful paradigm to tackle such sequential decision problems by learning hedging strategies from data or simulations. Moreover, adversarial reinforcment learning (ARL) brings a robustness perspective; the hedging agent is trained against an adversary (worst case scenario environment) that stresses the hedging strategy, thereby preparing it for regime changes and model misspecifications. This review provides a structured overview of these developments - from the foundations of option hedging theory to the latest adversarial RL approaches with a focus on simpler European options. We organise the narrative as follows: First, we recap traditional hedging models and identify their limitations under regime shifts. Next, we introduce RL in finance, tracing its evolution in option hedging and outlining key RL algorithms. We then explore the advent of adversarial and robust RL techniques for option hedging, explaining how they address model uncertainty and regime changes. A comparative synthesis of methodologies will be provided, including various quantitative metrics (hedging error variance, tail risk, Sharpe ratios, and training efficiency) to demonstrate progress over time. Finally we critically analyse current limitations from model risk and parameter sensitivity to training stability and discuss open challenges, linking these to the proposed theoretical foundations underpinning the solution in our chapter 3 "Problem Formulation and Theory".

2.2 Classical Approaches to Option Hedging

Modern option pricing and hedging theory began with the work of Black and Scholes (1973) and Merton (1973). In their frictionless market model with continuous trading, a derivative's price equals the cost of a self-financing replication portfolio (*see Appendix A for explanation of self-financing replication portfolio*) By continously rebalancing a portfolio of the underlying asset and cash, an option writer can delta-hedge so that the portfolio's value tracks the option's payoff, theoretically yielding zero hedging error *see Appendix B for explanation of Option Pay off.* Again, this relies on idealised conditions such as constant volatility, no jumps in the underlying's price, zero transaction costs, and the ability to trade continuously. Reality, however, dicates otherwise. Whilst theortetically possible, a trader cannot physically continuously hedge and market frictions means all risk cannot be eliminated. Despite these drawbacks, the Black-Scholes-Merton (BSM) model became the cornerstore of hedging practice, providing closed-form formulae for option prices under lognormal price dynamics. It established the paradigm that hedging is an optimisation problem: minimise the variance of the final hedging error, thus leading to more sophisticated models over time.

To relax the strict assumptions of BSM, later models introduced additional sources of uncertainty. Heston (1993) incorporated a stochastic variance process for the underlying asset, yielding a closed-form solution for European option prices with mean-reverting volatility. By introducing a stochastic process for volatility itself, it removes the assumption that volatility is constant. Other models such as Merton's jump-diffusion model (1976) allowed the underlying price to undergo occasional jumps. Other jump models have been introduced over time. Within our solution, we will focus on modelling the underlying using Heston and as such warrants an early close to exploration of additional simulation models.

In parallel to closed-form models, researchers applied stochastic control methods to hedging. Early on, the hedging problem was recognised as dynamic optimisation: choose a control such as hedge ratio to inimise the final shortfall varaince. This can be framed as a d ynamic programming problem. Local hedging approaches such as BSM cancel risk instantaneously by continuous rebalancing, whereas global approaches optimise the total risk over the life of an option. For example, Föllmer and Schweizer (1991) sought strategies that minimise the expected square hedging error at expiration, given discretre rebalancing. These problems often lead to recursive dynamic programs. Edirisinghe et al. (1993) and others cast hedging as a multi-stage stochastic linear program (a scenario tree), minimising shortfall across many simulated price paths. The optimal strategy can be viewed as a feedback control law: at each time, observe the current portfolio value and market state, then apply a control in the form of a trading action to correct any deviation from the desired hedging position. This control perspective treats market randomness as a disturbance to be rejected by appropriate feedback trading. The result of such efforts have confirmed that (i) perfect hedging is generally impossible in incomplete markets, and (ii) one must trade off risk and cost over the option's life, especially with regime shifts or structural changes occur. More importantly, these approaches assume a fixed probabilistic model for underlying dynamics (e.g. a known Markov regime-switching model), whereby the transition probabilities are known. For example, we would fix the probability for switching from a calm volatility regime to a stormy one. Given this assumption, if the model was wrong or regimes shifted unpredictably, the static strategy could perform poorly, highlighting a need for more adaptive and model-agnostic hedging methods.

By the early 2000s, the limitations of classical approaches under regime shifts were evident. Market data showed that volatility and jump intensity can change abruptly (e.g. during crises), breaking the staionarity assumptions of any single model. A strategy optimised for one regime (say low volatility) can incur large losses if a high-volatility regime arrives.Traditional hedging frameworks struggled with this: a BSM delta hedge, for instance, underhedges during volatility spikes because the model volatility is too low, leading to large shortfall losses. Even stochastic volatility models such as the industry standard Heston, if mis-calibrated or the volatility process itself shifts (e.g. a volatility of volatility shock), can leave hedgers exposed. Control-theoretic polices derived under one model can become suboptimal if the true dynamics deviate. In summary, what was learnt was to hedge under idealised or calibrated models and what emerged, was a realisation of the fragility of models under regime changes, parameter misspecifications, and other model risks. This set the stage for data-driven and learning-based approaches that could learn hedging strategies directly from market experience or simulations, and even prepare for worst-case scenarios. RL offers a way to do just that, by treating hedging as a sequential decision problem amenable to optimisation without the need to assume a fixed model.

2.3 Reinforcement Learning for Dynamic Hedging

RL considers an agent interacting with an environment to maximise a given cumulative reward. At any given time, the agent will be presented with a view of the world, known as an observation if the environment is partially observable, or a state if it is fully observable. Within the literature, for simplicity, it is often referred to as a state irrespective of observability. A state contains information about the environment. (*For readers less familiar with RL please see Appendix C for an overview of key concepts*)

In option hedging, the agent is the trader choosing how many shares (or other assets) to hold at each time, and the environment is the market that evolves stochastically. The state may include market features (underlying price, time left till expiry, volatility estimates, etc.), and the reward can be defined as the negative hedging loss or some risk-adjusted profit. Critically, hedging is a sequential decision task, exactly the setting RL is designed for. Without venturing into a detailed history of RL methods, a drawback has been difficulty in handling high-dimensional state spaces. Deep reinforcement learning (DRL) methods use neural networks as function approximators and have shown the ability to handle high-dimensional state spaces and complex dynamics that would foil traditional dynamic programming or partial differential equation methods. Buehler et al. (2019) was among the pioneers and coined the term "deep hedging" for using deep RL to learn optimal hedging strategies under realistic market conditions (including transaction costs and market impact). They applied a policy-gradient algorithm to directly minimise a risk objective (like CVaR) of the terminal hedging P&L. Around the same time, Halperin (2017) introduced the "QLBS" model, which framed the Black-Scholes world as a Markov Decision Process (MDP) and used Q-learning (a type of RL) to recover the optimal delta-hedging policy. These works opened the gates to research applying various RL algorithms to option pricing and hedging problems (see Carbonneau and Godin 2021; Cao et al. 2023; Mikkilä and Kanniainen 2023; among others). RL's appeal is that it can learn directly from data or simulated experience, potentially capturing complex patterns (nonlinear payoffs, path dependencies, regime shifts) that are hard to derive analytically. Moreover, once an RL policy is trained, it can adapt its hedging in real-time based on observed state variables (including features that can serve as proxies for regimes, such as volatility levels).

A variety of RL methods have been explored for dynamic hedging, each with strengths and considerations:

Value-Based RL (Q-Learning These methods focus on learning the value function, $Q(s, a)$, which estimates the expected cumulative reward (e.g. negative future hedging loss) from state s. In discrete action settings, the optimal policy is to choose the action with highest Q-value. Halperin's QLBS is an example that adapts Q-learning to continuous states (using regression to approximate Q) while discretising actions (e.g. small integer multiples of shares). Du et al. (2020) applied deep Q-learning (DQN) to hedge otpions,. treating the problem in a discrete action-space setting. A benefit of Q-learning is that it can leverage dynamic programming principles and is off-policy (can learn from any generated data). However, a major challenge is continuous action spaces. Hedging requires choosing a real-valued hedge ratio $h_t \in \mathbb{R}$, which is not naturally handled by DQN without discretisation. Discretisiation the action (e.g. number of shares) can introduce approximation error or huge action dimensionality. Extensions such as Continuous Q-learning i.e. using neural networks to output $Q(s, a)$ for continuous a can be used, but stability is not always guaranteed. Value-based methods also struggle with sparse rewards. In hedging the main payoff (reward) comes at option expiration, with maybe small costs in between. This sparsity can make temporal difference updates (which bootstrap on intermediate estimates) less accurate early in training.

Policy-Gradient Methods Policy-based RL directly parameterises the policy $\pi_\theta(a|s)$ (e.g. a neural network mapping state to a hedge action) and optimises the parameters θ to maximise expected reward or minimise some risk measure. Buehler et al. (2019) used a Monte Carlo Policy Gradient approach to minimise a risk objective (CVaR of the hedging P&L) rather than just expected loss. In Chapter 3 we will cover the benefits of minimising CVaR as a loss objective. Policy-gradient methods are well-suited to continuous action problems as the policy can output a continuous hedge ratio directly. They also allow optimising custom objective functions (not just expectation of reward) as briefly mentioned in the optimisation of CVaR. However, they require a lot of sample trajectories for gradient estimates (hgih variance of plain Monte Carlo gradients). Despite this drawback, in hedging, Monte Carlo gradients are actually quite effective as the reward is realised at the end, which aligns with episodic algorithms. In fact empirical studies suggest that pure Monte Carlo policy-gradient (only updating after final payoff) outperform temporally-smoothed updates in this context. This is because of the nature of option payoff, as temporal difference methods may propagate misleading intermediate values rather than Monte Carlo methods that capture the true outcome at the end.

Actor-Critic and Advanced Deep RL Actor-Critic algorithms combine the above two approaches: a Critic learns a value function, while an Actor adjusts the policy using feedback from the critic. Methods such as Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2016) and Proximal Policy Optimization (PPO) (Schulman et al., 2017) have been applied to option hedging. For example, Cao et al. (2020) and Marzban et al. (2023) use DDPG (continuous action actor-critic), and Du et al. (2020) experimented with PPO (discrete actor-critic). Actor-Critic methods leverage the best of both worlds: stable value estimation and continuous action support. PPO in particular was used to hedge options on a stock index, showing improvement over Black-Scholes in a volatile market scenario (Du et al., 2020). Actor-Critic methods benefit from greater sample efficiency than pure policy-gradient due to the existence of the Critic, and can use advanced tricks like advantage estimation and trust-region updates (PPO) for stability. However, they do add complexity through the introduction of many hyperparameters and moving partners e.g. DDPG must train two networks (Actor and Critic), tune learning rates for each, plus additional terms like exploration noise, target network update rate, etc.. In one study, a DDPG hedge performed marginally worse than a simpler DQN-based hedge, potentially because the added complexity and hyperparameters made training hard. Similarly, PPO can be sensitive to the choice of clipping parameter and learning rate. Overall such methods have shown promise, but their performance can heavily depend on tuning and the specifics of the market environment. We will delve into greater detail on PPO and its suitability for this problem statement in Chapter 3 and Chapter 4.

Model-Based versus Model-Free RL An important distinction in RL is whether one assumes a model of the environment dynamics. If model-based, some knowledge of the environment dynamics is assumed. One could use a known stochastic model such as Heston. Model-free on the other hand means the agent treats the environment as a black box and learns purely from observations (which can be simulated). Most applications in the literature are model-assisted in the sense that they simulate price paths from a model to train the RL agent (e.g. some two state Markov regime model and Heston model). The RL agent, however, does not necessarily know the model equations. It learns by trial and error on simulated data, making it effectively model-free in training. Truly model-based approaches are less common in option hedging given the risk of model misspecifications. Offline RL and Data-Driven Training A notable challenge in finance is that one cannot always train an RL agent online in the real market due to risk and lack of reset possibilities. Instead, there is reliance on simulated or historical data. Offline RL refers to training an agent purely on a fixed dataset of prior experiences without future interaction. This faces difficulty because standard RL algorithms assume they can explore and collect new data; offline RL must carefully avoid extrapolation beyond the data distribution. In hedging, pure offline RL is still nascent given most studies use simulated market models to generate as much training data as needed. Simulated market models enable researches to test various conditions, rather than relying on the limited financial markets history. However, some works have started evaluating policies on real market datasets. For example, Hirano et al. (2023) train their adversarial deep heding model (discussed in the following section) on actual historical price series and report performance comparable to model-driven methods. Offline or batch RL for hedging would aim to learn from historical option outcomes, perhaps using techniques like batch-constrained Q-learning or imitation learning from succesful hedging trajectories. A related concept is "Empirical deep hedging", where one uses real market data to both calibrate a simulator and test the learned hedging policy out-of-sample (Pickard & Lawryshyn 2023; Hambly et al. 2023 surveys discuss this). The key hurdle is generalisation. A policy trained on one regime might fail if a new regime not represented in the empirical data occurs. This motivates adversarial training, injecting hypothetical regime shifts during training, which we turn to next.

2.4 Adversarial and Robust Reinforcement Learning in Hedging

ARL introduces a second agent (often termed Nature of adversary. Within this thesis we refer to it as Nature) whose goal conflicts with the main agent's goal. Typically formualated as a zero-sum game, the protagonist (hedger) seeks to minimise some cost (hedging error) whilst the adversary seeks to maximise it. Pinto et al. (2017) first popularised Robust ARL in the context of robotics, showing that training an agent in the presence of an adversary applying disturbance forces can lead to polices that are robust to model perturbation and unmodeled dynamics. The adversary effectively generates "worst-case" conditions during training such as pushing a robot or in our case, pushign the market into difficult scenarios, so that the protagonist learns to handle them. In hedging, this idea naturally maps to robust hedging under model uncertainty; we can imagine ad adversary that tweaks the market's behaviour (within a plausible set) to hurt the hedger's performance. By training the hedging policy against this adversary, we hope the policy will perform well even in adverse or unexpected market regimes. Mathematically, one can frame this as a minimax problem: the hedger minimises a loss $L(\pi, P)$ while the adversary chooses market parameters P (e.g. transition probabilities, volatility, jump frequency) to maximise the same loss, within some feasible set $\mathcal{P}$ of models. Casting our mind back to Buehler et al. (2019) the hedger is minimises CVaR whilst Nature seeks to maximise it. The challenge is that solving this minimax exactly can be intractable for complex environments, so ARL uses simultaneous training: the adversary and agent update their strategies iteratively.

Financial applications of ARL have begun to appear in the last few years, driven by the need for robustness to regime shifts and model misspecification. One approach by Hirano et al. (2023), is Adversarial Deep Hedging. In their framework, a hedger network and a generator (market) network are trained against each other. The generator produces price paths of the underlying asset; it tries to fool the hedger by generating scenarios where the hedging strategy performs poorly, while still aiming to resemble realistic market behaviour. The hedger learns to hedge without assuming a specific price process model; effectively the generator is learning a worst-case model. This approach is inspired by Generative Adversarial Network (GANs), where the generator's outputs are constrained to look like real data (so that the adversarial scenarios are plausible, not completely arbitrary). The result is a hedging policy that is robust: it achieves low hedging error even when the actual underlying dynamics differ from any single assumed model. Notably, Hirano et al. report that their adversarially-trained hedger performed comparably to strategies that assume a correct model, across various real market datasets.

In a similar vein, Limmer and Horvath (2023) proposed Robust Hedging GANs. They explicitly address the problem of model uncertainty by automating robustification in the deep hedging framework. Their setup has three components: (i) a deep hedging engine (which could be a policy network), (ii) a data generating process), and (iii) a metric to measure discrepancy between the simulator's distribution and the real market distribution. The adversarial training comes in by penalising the generator for deviating from the reference distribution while it tries to worsen the hedger's outcome. In other words, instead of restricting the adversary to a narrow ambiguity set, they introduce a penalty for distribution shift so that the generator balances realism with adversarial intent. This prevents the adversary from proposing extremely unrealistic scenarios (something a worst-case mathematical solution might do) whilst allowing exploration of a wide range of adverse conditions.

Another approach to robust RL in hedging is to incorporate risk-aware objectives and ambiguity sets directly into the optimsation. For example, Wu and Jaimungal (2023) develop a robust risk-aware RL framework for option hedging. They use a policy-gradient algorithm with a robust performance criterion: the agent optimises some risk measure such as CVaR while considering that the data-generating process might vary. In their work on hedging barrier options, they examine how the optimal policy changes as the agent becomes more risk-averse and how the robustified strategy outperforms a non-robust one when the test market dynamics differ from the training dynamics. This highlights that robust RL need not always involve an explicit adversary network as one can solve the inner maximisation analytically or via simulations. In some cases, if the uncertainty in the model is parameterised simply (say by a set of alternative volatility values or alternative transition probabilities in a Markov chain), one can enumerate the worst case. This kind of robust MDP formulation (see Iyengar, 2005; Nilim & El Ghaoui, 2005) provides theoretical guarantees: the hedging policy is optimal against all transition matrices in the uncertainty set. The trade-off is that the uncertainty set must be specified in advance and kept relatively simple. A benefit of this, is that training two agents (hedger and adversary) is computationally intensive and can be unstable. The minimax game may not converge smoothly and can oscillate if the adversary and protagonist keep leapfrogging in performance.

2.5 Comparative Analysis of Hedging Strategies and Performance

We can see a variety studies have been undertaken, giving us the ability to compare tradition hedging, basic RL hedging and robust/adversarial RL hedging on common metrics. We briefly synthesize some key findings below focusing on quantitative performance measures:

Hedging Error Variance A primary metric is the variance of the hedging P&L error at maturity. Buehler et al. (2019) reported their deep hedging policy (trained to minimise CVaR) achieved notably lower variance in terminal hedging losses compared to a delta-hedge, especially with markets with jumps and transaction costs. More systematically, Neagu et al. (2024) compared eight different DRL algorithms against the Black-Scholes delta hedge baseline on a simulated market. They found Monte Carlo Policy Gradient was the only RL algorithm to significantly outperform the delta hedge in terms of hedging risk, give the same computational budget. Other more sophisticated algorithms such as PPO, DQN variants, achieved variance on par or slightly above the dleta hedge suggesting that not all RL hedges are automatically better, and it it depends on the algorithm and objective used.

Tail-Risk Measures (CVaR and VaR) Focusing only on variance above can be misleading if hedging errors are skewed or fat-tailed. Many recent works therefore optimise and evaluate Conditional Value-at-Risk (CVaR) of the hedging loss distribution. CVaR_α measures the average loss in the worst $(1-\alpha)$ of cases, effectively a tail average. Carbonneau and Godin (2021) introduced an equal risk pricing framework using deep hedging, where the rpice of an option and hedging strategy are determined such that the CVaR of the short position's loss is minimised as a certain confidence level. By doing so, they ensured the hedger's tail risk was controlled and their deep RL strategy had much lower CVaR at 95% than a Black-Scholes delta hedge priced at the same premium. We will see more detailed treatment of CVaR and its significance in Chapter 3. In general, deep hedging strategies tend to dramatically reduce tail risk: studies have shown 30-50% reductions in worst-case losses relative to naive hedging in some scenarios (e.g., during 2008-type volatile periods, according to Horvath et al. 2021). The CVaR is commonly used as an objective in deep hedging literature and tuning the confidence level α gives a risk/return trade-off. A higher α forces the strategy to focus on extremely bad outcomes,. often yielding a more coinseravtive hedge. Conversely lower α may allow more aggressive strategies that accept some tail risk in exchange for lower cost. Crucially, adversarial and robust RL methods shine in improving tail metrics: Wu & Jaimungal (2023) demonstrated that a robust risk-aware strategy had a much tighter loss tail than a standard risk-neutral strategy when the test data generating process different from the training assumption, evidencing that when faced with 'surprising' scenarios ARL methods outperform on this metric; crucial for risk management.

Other metrics Other metrics such as Sharpe Ratio and Risk-Adjusted Return show that there are lower absolute returns on average due to the conservative approach but when accounting for risk perform well. In addition it is informative to compare on computational criteria. Unsurprisingly, the more complex some algorithms are, the longer they may take to converge, although this does not necessarily mean that they perform worse; simply from a computational standpoint they are more expensive. This indicates that there is a trade-off to be had, where more complex algorithms can yield better results, but they demand more data, compute, and careful tuning. Simple methods may converge fast but could plateau at a suboptimal hedge. Therefore the frontier of research is pushing for methods that are both sample-efficient and robust, such as improving training with gradient estimators or incorporating domain knowledge.

2.6 Limitations and Open Challenges

Despite significant progress, several limitations and open issues remain in robust option hedging, especially under regime shifts and with adversarial training.

Model Ambiguity and Regime Change Although adversarial RL can provide robustness to modeled regime shifts, it is only as good as the assumed set of scenarios. In practice, truly unanticipated regime shifts (e.g., the COVID-19 market crash, which may not resemble any scenario in the training data or uncertainty set) can still lead to large losses. It is difficult, therefore to guarantee hedging robustness against every possible future regime, only those we contemplate. The hope is that by training on a wide variety of stressed scenarios, the policy generalises to unseen ones, but there is no certainty. Additionaly many ARL models assume the regime is unobservable (a hidden Markove chain). The hedging agent then relies on filtering or inference (e.g. the belief q_t of being in a high volatility regime). If the inference is wrong or slow, losses can accumulate. Robust hedging under partial observability is a complex challenge, and simplifications such as assuming the filtering is given or incorporating the belief state into the RL state are done to combat this.

Parameter Misspecification and Calibration A persistent issue is how to choose the uncertainty set for robust hedging. If too narrow, the adversary may not prepare the hedger for what actually happens. Too board, and the worst-case may be overly pessimistic and lead to overly costly hedging. Therefore, the trade-off between robustness and cost requires careful consideration. Considering how adversarial is "too adversarial" remains somewhat subjective. A baseline model to generate training scenarios is an important tool and must be calibrated well.

Risk Measure Sensitivity and Objective Alignment As highlighted by Gauthier and Godin (2024), the choice of risk measure in the objective can fundamentally alter the hedging policy. Optimising for profit may lead to unnecessary risk, whereas optimising a very tail-focused measure like CVaR can make the strategy overly conservative. It ultimately depends on how the objective is set up and the risk appetite of the practitioner at hand.

Training Stability and Robustness of Learning Adversarial training introduces potential instability. There is a risk of mode collapse, where the hedger learns to handle one worst-case scenario, then must adjust to another worst-case, running the risk of oscillating without convergence if learning rates are not well-tuned. The challenge of ensuring the training does not overfit toa particular stress scenario also remains.

Evaluation and Interpretability Finally, evaluating a robust hedging strategy in real markets is challenging. The lack of closed-form benchmarks for performance under regime uncertaintiy means reliances on simulation or historical backtesting. A strategy that performs well in simulation may underperform in reality. Additionally, RL policies, particularly neural network ones, are black-boxes and interpreting what the policy has learned is difficult but important for trust and adoption. Interpretability is not a core focus on current literature and will not be looked upon in this thesis, but remains a key consideration for future works particularly if RL methods are to make their way into industry.

2.7 Where we stand now

Robust and adversarial reinforcement learning has shown proof-of-concept success in improving option hedging under regime shifts and model uncertainty. Policies exist that can beat classical delta-hedging in difficult environments, and the methodology to train such policies (whether via two-player games or risk-augmented objectives) is established. However, what remains unresolved is unifying these approaches into a coherent framework that can handle the full complexity of real markets (multiple regimes, partial observability, transaction costs, etc.) with guaranteed stability and performance. This thesis will contribute to this frontier by providing a specific formulation for regime-switching hedging with an adversarial twist (worst-case Markov chain dynamics) and deriving theoretical insights (like the nature of the worst-case "corner" regimes). This helps bridge the gap between theory and practice by simplifying the adversarial problem to something tractable and explainable. The end goal is a robust, adaptive hedging framework that can reliably manage risk through unforgiving market regimes.

3 Problem Formulation and Theory

Given our understanding of the literature up to date and some of the constraints and tools available to us, we can look to formally formulate the problem and discuss the theory underpinning the proposed approach. Prior to this section, it would be prudent for the reader to browse through *Appendix A* should they require some intuition on a self-financing portfolio.

3.1 Market and Hedge Dynamics

We model an option trader who has sold an option and must dynamically hedge the position by trading the underlying asset. The goal is to minimise the final hedging shortfall when the option expires. We first look to discretise the trading horizon $[0, T]$ into N equal steps of length $\Delta t = \frac{T}{N}$. Let S_t denote the underlying price and C_t the option fair value at time t.

3.1.1 Underlying dynamics

At each step:

$$S_{t+\Delta t} = S_t \times \exp[(\mu - \frac{1}{2}\sigma_t^2)\Delta t + \sigma_t \sqrt{\Delta_t} Z_t]$$

where $Z_t \sim \mathcal{N}(0,1)$ and the latent volatility regime σ_t takes the low or high value as defined in section 3.2. We note that the drift term contains $-\frac{1}{2}\sigma^2$, the Itô correction, because the exponential map is convex (the second derivative is always positive) and in stochastic calculus, the quadratic variation satisfies $dW_t^2 = dt$ rather than zero as in deterministic calculus. This term subtraction removes the "convexity lift" implied by Jensen's inequality. It guarantees that under $E[\cdot]$ the discretised process has mean growth $e^{\mu \Delta t}$, keeping the log-normal model martingale under the risk-neutral measure.

3.2 From random walks to the Heston model

We now discuss how to simulate the price of an underlying asset and why such mathematical models are necessary. At its core, hedging depends on our ability to predict how derivative securities will respond to changes in underlying asset prices, market volatility, to name a few. Therefore, mathematical models are required that can capture the essential statistical properties of asset price movements with sufficient accuracy. Real world observations in many cases, not just asset prices, exhibit randomness. If asset prices followed predictable patterns, arbitrage opportunities, that is, the ability to purchase something and immediately sell it for a profit, would be immediately exploited given sufficient capital. Consider an equity price chart, where every tiny tick is buffeted by a constant stream of influences that no deterministic model could ever capture. News arrives unpredictably, traders receive private information, and market microstructure effects create noise that persists across multiple time scales. We can use a mathematical framework to capture this essential randomness by looking at the percentage change as

$$\frac{dS_t}{S_t} = (\text{predictable drift})dt + (\text{unpredictable volatility})dW_t]$$

where W_t represents standard Brownian motion; a continuous-time mathematical representation of pure randomness. There are several key properties that make Brownian motion the right tool for modeling financial uncertainty.

1. $W_o = 0$ to provide a clean reference point for measuring cumulative random effects.

2. In any tiny interval Δt, the random increment $dW_T \sim \mathcal{N}(0, \Delta t)$ captures the idea that uncertainty accumulates gradually.

3. Increments over non-overlapping time periods are completely independent, reflecting the efficient market hypothesis that past price movements provide no information about future changes.

What makes the Brownian motion different is its quadratic variation: $dW_t^2 = dt$ rather than zero as in deterministic calculus. This means even infinitesimal random movements accumulate measurable effects over time, which is why stock price models require stochastic treatment.

We model the percentage change as

$$dS_t = \underbrace{\mu S_t dt}_{\text{drift term}} + \underbrace{\sigma S_t dW_t}_{\text{diffusion term i.e. volatility or "shake"}}$$

We care for percentage changes because absolute movements have different impacts depending on the existing price. A \$2 move on a \$10 stock means more than a \$2 move on a \$600 stock. When solving this SDE, we obtain the result in 3.1.1.

The Geometric Brownian Motion has a strong assumption that violates that which we are seeking to solve. It assumes volatility σ remains constant through time. We, however, are aware that markets exhibit regimes whereby volatility itself can shift. Upon spectating the market, one will observe distinct behaviour regimes: extended periods of relative calm, perturbed by sudden volatility explosions seemingly without warning. Whilst models require assumptions to work, given all models are an approximation of reality, we are able to fine tune the Brownian Motion idea to attain a model that exhibits a more accurate representation of the dynamics we seek to solve, specifically random behaviour in volatility.

Given volatility exhibits random behaviour, it would be prudent to allow σ itself to evolve stochastically rather than remaining frozen at some historical average. We use the Heston (1993) model by treating variance as its own mean-reverting.

$$dS_t = \mu S_t dt + \sqrt{v_t} S_t dW_t^S$$

$$dv_t = \kappa(\theta - v_t)dt + \xi\sqrt{v_t}dW_t^v$$

There are 5 parameters:

1. v_0 : initial variance
2. θ : the average variance of price over some period. $\mathbb{E}[v_t]$ will tend to θ as t tends to ∞
3. p : the correlation of the two stochastic processes provided
4. κ : a mean reversion term i.e. the rate at which v_t reverts to θ
5. ξ : the volatility of the volatility which determines variance of v_t

If the following condition $2\kappa\theta\xi^2$, the process v_t is strictly positive. In the following section, we discuss behavioural shifts between well-defined regimes. Whilst Heston model volatility as a continuous diffusion process, we capture the real world economic insight that markets will alternate between volatility states with distinct discrete parameters. Market participants naturally think in terms of market regimes rather than continuous stochastic processes. In practice we will keep the Heston form for both regimes but assign low versus high parameter sets, letting the Markov chain decide when parameters switch. We will see that this discrete space enables our minimax optimisation framework while maintaining computation tractability.

3.3 Market regime and uncertainty sets

3.3.1 Markets have moods

Markets have moods. Most days an equity index drifts with some noise; occasionally it erupts into violent swings. Volatility is therefore stateful; the market switches between calm and storm moods. We capture these moods with an unobservable two-state Markov chain. As a result, the agent must hedge without seeing the regime directly. Before we continue, let us briefly discuss key terms.

1. Markov Chain: a stochastic process whose next state depends only on the current state
2. Hidden Regime note: the latent volatility state σ_t; the agent works instead with a filtered belief $q_t = Pr(\sigma_t = \sigma_H|\mathcal{F}_t)$ i.e. the probability volatility is in a stormy state given information up until time t

A finite-state Markov Chain moves between states according to some transition matrix whose rows are probability vectors (must sum to 1). We denote the latent volatility state at trading dat t by:

$$\sigma_t \in \{\sigma_L, \sigma_H\}$$

where

1. σ_L = [8%] represents the "calm" annualised volatility level
2. σ_H = [21%] represents the "storm" annualised volatility level (typically 3x σ_L)

We encode the one-step dynamics by:

$$P = \begin{pmatrix} p_{LL} & 1 - p_{LL} \\ p_{HL} & 1 - p_{HL} \end{pmatrix}$$

State L indicates a low volatility day, thus state H indicates high. Calm and storm. The four entries of matrix P tell us how sticky each regime or mood is. Note that given rows of P sum up to 1. P_{LL} is the probability that a low volatility regime remains so and $1 - P_{LL}$ is the probability of a switch to a high volatility regime occurring. The second row is interpreted analogously.

Classifying daily SPX returns from 2010 to 2024 into calm or storm regimes yield the empirical estimate, rounded to two decimal figures

$$\bar{P} = \begin{pmatrix} 0.98 & 0.02 \\ 0.04 & 0.96 \end{pmatrix}.$$

These numbers line up with earlier regime-switch studies (Hamilton Susmel 1994; Guo 2013). Whilst these numbers are slightly different, they are within and band.

3.3.2 Addressing model risk through uncertainty sets

Historical calibration provides our best estimate of regime transition behaviour, however, it remains imperfect; a new crisis could shorten spells or lengthen storms. Market conditions change over time, sample periods may not capture all possible regime dynamics, and the estimation error creates additional uncertainty around the true transition probabilities. To protect against such model risk we give Nature (adversary Ω) the right to nudge any entry of the matrix by at most ϵ whilst preserving each row sum. This uncertainty set captures our acknowledgment that the true regime dynamics may differ systematically from our calibrated model.

$$\boxed{\mathcal{P} = \left\{ P : \|P - \bar{P}\|_\infty \le \varepsilon,\ P\mathbf{1} = \mathbf{1} \right\}} \qquad \varepsilon = [\mathbf{0.02}].$$

The parameter epsilon represents our level of confidence in the historical calibration. Larger epsilon values create more conservative hedging policies that aim to perform well across a broader range of possible regime dynamics, whilst smaller values produce policies more closely tailored to the historical estimate. Smaller epsilon values result in hedging policies that perform well when historical calibration proves accurate but potentially perform poorly when market conditions differ significantly from the training period. Larger epsilon values create hedging policies that maintain reasonable performance across a much wider range of market conditions, however, optimality is sacrificed when historical estimates prove accurate. This connects to the classic bias versus variance trade-off. Thus, epsilon selection requires careful thought; the most natural approach is based on the statistical uncertainty inherent in the parameter estimation process. With all historical data and associated statistical estimates, they come with confidence intervals that reflect the precision of our estimation procedure. We base the epsilon on the Wilson 95% confidence interval for each entry of $\bar{P}$. Wilson's rule is a statistically safer margin of error for proportions. Taking the largest half-width across all four entries gives a data-driven bound ϵ_{stat} - 0.015; we round up to 0.02 for a modest safety margin. We introduce the infinity-norm (largest value) that acts entry-by-entry:

$$\|P - \bar{P}\|\infty = \max i, j|p_{ij} - \bar{p}_{ij}|$$

Saying this quantity is $\leq \epsilon$ is equivalent to putting the same absolute tolerance $\pm\epsilon$ for each transition probability. Graphically, the admissable points form an axis-aligned rectangle around $\bar{P}$ in the $(P_L L, P_H L)$ plane. In practice, this provides several benefits:

1. The desk can think about "How wrong might the calm-to-storm probability be" without simultaneously worrying about the storm-to-calm leg; each has its own $\leq \epsilon$ band

2. One can easily move transitions to their lower and upper bands if choosing to shorten calm spells and lengthen storm spells. Both stresses provide simple explanations to non-quant stakeholders, because only one intuitive metric is changing at a time.

3. A rectangle is convex, so the robust MDP machinery guarantees the minimax value is reached by a deterministic stationary policy and admits efficient dynmaic-programming evaluation,. This provides tractability.

We will briefly touch on the point 3, which is the significance of convexity. $\mathcal{P}$ is an axis-aligned rectangle, hence convex. The convexity property provides essential because it triggers the existence guarantees established by Iyengar (2005) for robust Markov Decision Processes. When uncertainty sets are convex and satisfy additional regulatory conditions, deterministic stationary minimax policies exist and can be computed using standard dynamic programming techniques as noted above. Looking at our hedging application, we want to find policies that minimise the worst-case performance over all possible transition matrices in the uncertainty set. Suppose we lack mathematical guarantees, such problems may yield no solutions, multiple solutions, or solutions that are too computationally expensive to obtain.

3.3.3 Game structure and information flow

$$\boxed{\Omega : P \in \mathcal{P} \implies \sigma_t \xrightarrow{\text{Heston}} S_t \xrightarrow{\pi} h_t}$$

At the start of an episode, Nature Ω secretly draws a matrix P inside the ϵ rectangle. Within the daily loop, the hidden chain will generate σ_t which drives price dynamics. The agent observes prices, updates belief q_t and sets hedge $h_t = \pi(X_t)$

3.4 Conditional Value at Risk (CVaR

3.4.1 Why tail risk matters

A delta-hedged option book is usually sleepy; 95% of the time, Profit and Loss (PL) wiggles within a narrow daily band. Most trading days follow predictable patterns, most market movements fall within expected ranges, and most hedging strategies work reasonably well under normal conditions. Blow-ups happen in the last 5% of days including but not limited to earning calls, flash crashes, pandemics. If risk metrics ignore the magnitude of these bad days, significant losses can occur.

This asymmetry between frequency and impact becomes particularly acute in regime-switching environments. During calm periods, hedging errors remain manageable and traditional risk measures provide reasonable guidance. However, when markets transition storm spells, the distribution of potential losses develops much fatter tails, resulting in greater probability mass in extreme loss regions. This leaves us with strategies generating errors precisely when they are needed most. hence the focus of modern trading desks has shifted to tail risk, hence our robust hedging framework should optimise a statistic that cares about the size of the tail, not just its probability.

3.4.2 Definition of Conditional Value at Risk

For a loss random variable L and confidence level α (assume 95%), the Value at Risk is defined as:

$$VaR_a(L) = inf\{\ell : Pr[L \leq l] \geq \alpha\}$$

This tells us the smallest loss threshold to cover 95% of all possible days (assuming α is 95%). (*Please see Appendix Dfor a breakdown of VaR.*) This is useful, however, it fails to tell us by how much we breach the candidate breach value. CVaR accounts for this by examining what occurred in $1 - \alpha$ of cases.

$$\boxed{\mathrm{CVaR}\alpha(L) = \frac{1}{1-\alpha}\,\mathbb{E}\big[L \bigm| L > \mathrm{VaR}\alpha(L)\big]}$$

To put it simply, it calculates the average loss in the worst 5% of scenarios given an α of 95%. Consider a scenario where 95% VaR is \$1m but 95% CVaR is \$4m. The tail is fat; most of the bad 5% days are far worse than \$1m.

3.4.3 Why CVaR dominates VaR for hedging

Captures severity not just cut off We see above with the example of \$1m VaR the benefits of CVaR. Assume two trading books have identical VaR but wildly different CVaR. Only the latter will flag which book will explode.

Coherent and convex (Artzner et al., 1999) CVaR satisfies sub-additivity, meaning that diversifying portfolios never increase CVaR. This proper aligns with the economic intuition that one can reduce, but never increase risk through diversification. VaR can actually increase after diversification due to mathematical pathologies that make it unsuitable for portfolio optimisation. The coherence properties ensure that CVaR provides economically sensible guidance for risk management decisions.

Optimisation friendly Rockafellar Uryasev (2000) showed that CVaR can be written as the minimum of a convex objective, so gradient based algorithms and dynamic programming recursion apply without hacks.

3.4.4 CVaR under model-uncertainty

Our hedge problem is a two-player game. The agent has a policy Π which determines how many shares to hold each day. It seeks to minimimse the CVaR. Nature Ω, our adversary, cho;oses a transition matrix $P \in \mathcal{P}(\epsilon)$. It seeks to maximise this same CVaR. Given Nature moves after seeing our policy, we must prepare for the worst-case CVaR. The reader may question why do we play a game whereby Nature can observe our strategy and then create its own. The information asymmetry can seem almost unfair and not reflect reality. However, the objective is to create a policy by which our agent can weather extreme events. It is akin to crafting points in a debate that your opponent is privy to. We must ensure no matter how strong our opponent's move or rebuttal is, our policy is robust.

Convexity is key here. Firstly, consider the possibility that our objective function is not convex and exhibits multiple local minima scattered throughout the parameter space. Our optimization algorithm may converge to one of these local minima and report that the optimal solution has been obtained, when, in reality, a better solution exists elsewhere.

Consider another circumstance where our objective function exhibits discontinuities of undefined regions. Such instances can cause our algorithm to crash, produce inconsistent results, or converge to points that represent mathematical artifacts rather than economically meaningful solutions. Such cases can waste significant computation time whilst simultaneously yielding unreliable results.

Suppose we have two policies

1. Policy A involves aggressive position adjustments that produce a CVaR of 2 million under certain market conditions

2. Policy B involves conservative position management that produces CVaR of 4 million under the same conditions

Assume we now flip a coin between these two strategies. Convexity property guarantees the mixed strategy will have a CVaR between 2 million and 4 million. Even if we opt for an inferior strategy it cannot be worse than 4 million. If we go 50/50, we observe a CVaR of 3 million. Whilst these seems trivial and almost obvious, not all mathematical functions operate like this. Thus for any $\lambda \in [0, 1]$

$$\mathrm{CVaR}\alpha\big(\lambda L_A + (1-\lambda)L_B\big) \;\leq\; \lambda\,\mathrm{CVaR}\alpha(L_A) + (1-\lambda)\,\mathrm{CVaR}_\alpha(L_B).$$

Returning to the uncertainty set, we note that each point in the uncertainty set represents a different set of regime dynamics that Nature may choose. Intuitively, Nature's choice could be anywhere. However, given CVaR is convex, any interior point, which itself is a weighted average of the extreme corner points, will not exceed the corner points. Therefore points will be at least as large as any interior point, ergo Nature, which always seeks to maximise this objective, will select corner points. This means that Nature needs to only enumerate through four specific transition matrices changing this from a continuous optimisation problem to discrete, significantly reducing computation load.

3.4.5 Economic Interpretation

Each corner of the uncertainty set corresponds to some stress scenario rather than some abstract mathematical construction.

1. Bottom left corner reflects calm periods are less likely to persist and storm days are most likely to persist

2. Top right corner reflects the opposite, where calm periods increase and storm periods are more likely to change to calm

3. The other corners give mixed cases capturing realistic possibilities where different market conditions impact differently.

3.5 The Minimax Objective

Now we are able to write the complete mathematical formulation of our robust hedging game. Our agent, as discussed above, seeks a policy to minimise the worst-case CVaR across all possible regime dynamics that Nature might choose. We denote the minimax objective objective in the Rockafeller-Uryasev (hinge) form:

$$\boxed{\min_{\pi \in \diamond}\;\max_{P \in \mathcal{P}(\varepsilon)}\;\min_{\zeta \in \mathbb{R}}\Big\{\zeta + \frac{1}{1-\alpha}\;\mathbb{E}\tau \sim (\pi, P)\big[(L(\tau) - \zeta)_+\big]\Big\}}$$

1. Agent (policy π) chooses hedge strategy

2. Nature Ω picks the worst regime within the uncertainty set $\mathcal{P}(\epsilon$

3. Buffer ζ self-adjust to the VaR level, converting the tail-average CVaR into a objective *Please see Appendix E for a detailed breakdown on self-adjustment to VaR*

We note the inclusion of a new buffer variable ζ. As we know, with calculating CVaR, we collect all possible loss outcomes, sort them, find the percentile that matches our α and average everything above that threshold. However, sorting and finding percentiles brings its own set of problems such as being inherently discontinuous. We must consider the issue of computing gradients when changing the policy and reordering the loss distribution (*Please see Appendix F for a breakdown of the issues that arise on this matter*)

3.5.1 Swapping Min and Max

From the hinge form formula, we note the computationally challenging nested sstructure because for every π we consider, we must solve an inner optimisation problem to find the worst-case P and optimal ζ. If we could swap the P and ζ around, we could solve for the policy and threshold jointly first, then find Nature's response. This much more tractable algorithmically, as we would not need to solve Nature's problem as a subroutine within each iteration of our policy search. Fortunately, we can do this because for any fixed policy π and threshold ζ, the objective function $\zeta + \frac{1}{1-\alpha}\mathbb{E}[(L^{\pi}(P) - \zeta)_+]$. To explain linearity in P, suppose we have our transition matrix and we wish to calculate the expected loss over a two-day period from a low volatilty state. Observe the four outcomes

$$\begin{aligned} \text{Low} \to \text{Low} \to \text{Low}: &\quad P_{LL}\,P_{LL} \\ \text{Low} \to \text{Low} \to \text{High}: &\quad P_{LL}\,P_{LH} \\ \text{Low} \to \text{High} \to \text{Low}: &\quad P_{LH}\,P_{HL} \\ \text{Low} \to \text{High} \to \text{High}: &\quad P_{LH}\,P_{HH} \end{aligned}$$

Suppose now, we have found our hedging policy π and threshold ζ for this example. Each of the four trajectories has a determined loss. P only changes how likely a path is, not the dollar attached to it. Let us now say, the losses are \$1.2m, \$2.8m, \$3.5m, and \$4.9m respectively. Observe the objective fuction

$$\mathbb{E}[(L^{\pi}(P) - \zeta)] = \underbrace{1.2}_{\text{fixed}} \times \underbrace{(P_{LL} \times P_{LL})}_{\text{can change}} + \underbrace{2.8}_{\text{fixed}} \times \underbrace{(P_{LL} \times P_{LH})}_{\text{can change}} + \underbrace{3.5}_{\text{fixed}} \times \underbrace{(P_{LH} \times P_{HL})}_{\text{can change}} + \underbrace{4.9}_{\text{fixed}} \times \underbrace{(P_{LH} \times P_{HH})}_{\text{can change}}$$

Since weighted sums are linear in their weights, this expectation is linear in P, which is precisely the property required by Sion's minimax theorem to justify swapping the optimisation order. This linearity holds irrespective of the number of market states or trading horizon. With this in mind, we now obtain a more tractable form where we solve for the policy and threshold jointly, then find Nature's worst case response.

$$\boxed{\min_{\pi \in \diamond} \min_{\zeta \in \mathbb{R}} \max_{P \in \mathcal{P}(\varepsilon)} \Big\{ \zeta + \frac{1}{1-\alpha} \, \mathbb{E}\tau \sim (\pi, P)\big[(L(\tau) - \zeta)_+\big] \Big\}}$$

3.5.2 Nature's simplified problem: The Extreme Point Theorem

For a fixed (π, ζ), the maximiser

$$P^* = \arg \max_{P \in \mathcal{P}(\varepsilon)} \mathbb{E}[(L^\pi(P) - \zeta)_+]$$

is a vertex of the $\epsilon-$recntangle $\mathcal{P}(\epsilon)$. Note from section 3.4, Nature need only evaluate the objective function at the four specific corner matrices and select the transition matrix which maximises CVaR. For any candidate π, ζ we can therefore enumearte these four matrices, compute CVaR exactly, and pick the worst case scenario without numerical tolerance issues. This discrete step is what makes the robust PPO algorithm that will be discussed in Chapter 4, both fast and exact.

3.5.3 Computational structure and implementation

Each training epoch repeats three steps. We will delve into this in more detail in Chapter 4, when we discuss methodology. For now an overview is

1. Nature enumerates the four rectangle corners and picks P^*
2. Buffer update: $\zeta \leftarrow \zeta - \eta_\zeta \Big[1 - \dfrac{1}{1-\alpha}\, \widehat{\mathbb{P}}(L > \zeta)\Big]$ whose gradient vanishes exactly at VaR_α
3. Policy update: apply PPO to minimise the tail loss $(L^\pi - \zeta)_+$

3.6 Existence and stationarity

Before training an agent two guarantees are required

1. Existence: the minimax CVaR value must be finite and attainable
2. Stationarity: an optimal policy can be kept time-homogenous i.e. one deicision rule $\pi^* : S- > A$

It is important to note that time to expiry is included int eh state space and as such we maintain a finite state set and do not violate Iyengar's theorem. Our neural network, architecture discussed in Chapter 4, will recieve "time till expiry" as an input feature

3.6.1 The three pillars of Existence

Because the state and action spaces are finite, the uncertainty set $\mathcal{P}(\epsilon)$ is a compact rectangle, and the objective is convex-concave, the robust MDP reuslts of Iyengar (2005) and Nilim El Ghaoui (2005) apply. They guarantee that an optimal stationary policy π^* and a saddle point (π^*, ζ^*, P^*), and that policy-gradient methods converge to this minimax value.

Finite State and Aciont Spaces Consider a function $f(x) = x$ on the real line,. Assume we sought the maximum value; there is not one since we can always choose a large x giving us an unbounded function. The finite number of volatility regimes prevent this unbounded issue. In finite spaces, continuous functions will always attain their maximum and minimum values, therefore we will not have situations where we escape to infinity.

Compact Uncertainty Set Given $\mathcal{P}(\epsilon)$ is a compact rectangle, it holds two properties

1. Closed: contains all the boundary points. This means we do not asympottocially approach the boundary points. We have the Extreme Point Theorem where maximum and minimum values are attained.
2. Bounded: Does not extend to infinity in any direction.

We know our uncertainty set is defined by $\|P - \bar{P}\|_\infty \leq \epsilon$ which creates a rectangular region around the calibrated transition matrix $\bar{P}$. It is bounded as entries cannot deviate more than ϵ and closed as it includes the boundary where deviations equal ϵ. Since continous functions can achieve these minimum and maximum deviations, it guarantees Nature can find the worst-case transition matrix P^*

Convex-Concave Structure Our objective function has a special mathematical structure

1. Convex in (π, ζ) : Agent's variables
2. Concave in P : Nature's variables

This gives us Sion's theorem that given these properties

$$\min_{x \in X} \max_{y \in Y} f(x, y) = \max_{y \in Y} \min_{x \in X} f(x, y)$$

which simply tells us, the worst-case scenario that Nature can create against our best strategy is the same as the best strategy we can deploy against Nature's worst case. This is why it fundamentally does not make a difference that Nature goes second after seeing our move.

3.6.2 Why Stationarity emerges

In our setup we posit the following:

1. Markov property: future regime transitions depend only on the current regime, not the entire history. This may seem counterintuitive given the assumption that calm days are though to naturally follow calm days, however, it simply means tomorrow depends only on today and the day after will depend only on tomorrow.

2. Finite Horizon: the episode ends at option expiration
3. Stationary environment: the transition probabilities do not change over time

These conditions mean that the optimal decision at any state does not depend on calendar time, only the current market state, despite of course time to expiry being a feature. This is precisely why we are able to use a stationary policy π^* rather than a time-dependent policy $\pi_t^*(s_t)$

3.6.3 Practical implications

These theoretical guarantees have direct computational benefits:

1. Our RL algorithm is guaranteed to converge to the global optimum
2. We benefit from simpler stationary policies rather than complicated time-dependent ones, therefore we do not require time-indexed policy weights resulting in fewer parameters
3. The minimax value provides a meaningful worst-case performance bound ensuring robustness

3.7 Algorithm Choice

Having established our minimax CVaR objective and proven the existence of optimal stationary policies, we now face the practice question: which learning engine can shoulder all four structural challenges?

1. continuous hedge ratios
2. hidden (unobservable) volatility regime states
3. tail risk (CVaR) focus
4. adversarial dynamics (corner-matrix Nature)

As such we require a practical RL engine that can accomplish the following

1. Discover that policy in a continuous hedge space
2. Cope with hidden-regime uncertainty
3. Align neatly with the CVaR-buffer objective whilst remaining stable under adversarial stress

3.7.1 Why Policy Gradient Methods

The nature of our hedging problem rules out classical value-based RL approaches such as Q-learning or DQN. At each trading period, our agent must choose a hedge ratio $h_t \in \mathbb{R}$, representing the number of shares to hold. This continuous action space creates an infinite dimensional optimisation problem that traditional tabular methods cannot address.

One may argue to discrete the action space into bins (e.g. hedge ratios from -2 to +2 in increments of 0.1), however, it introduces constraints that may be costly. We may see a situation where a hedge ratio of 0.45 may be optimal, but forcing the agent to choose between 0.4 and 0.5 could lead to significant errors over time. Additionally, to even achieve acceptable performance, we would have to implement such a large discrete action space which would defeat the purpose of discretising in the first place.

Policy gradient methods elegantly sidestep this issue by directly parametrising a stochastic policy $\pi_\theta(h|s)$ that outputs a probability distribution over continuous hedge ratios. During training, the agent samples from this distribution, naturally exploring the continuous space while gradually concentrating probability mass around optimal actions as learning progresses. This in turn, also eliminates the need for tuning exploration decay as seen in classical algorithms.

3.7.2 Which Policy Gradient Method?

Proximal Policy Optimisation Schulman et al. (2017) introduced Proximal Policy Optimisation (PPO) which takes a conservative approach to policy updates. By constraining each update to remain within a "trust region" of the previous policy, resulting in a clipped objective. Within our use case, assume Nature discovers a particularly devastating transition matrix that causes large losses. A naive policy gradient might overreact, drastically changing the hedging strategy based on what could have been a rare event. PPO's clipping mechanism acts as a natural damper, ensuring that even under adversarial pressure, policy updates remain measured and stable.

Before continuing, let us visit other possible choices and why they have not been selecting. Trust Region Policy Optimsation (Shulman, et al. 2015) works similar to PPO utilising Kullback Leibler (KL) Divergence, however, it ituilises second-order derivatives which adds significant complexiy, despite techniques such as calculating the conjugate gradient. Empirically PPO has attained the same KL control as TRPO.

Soft-Actor-Critic (SAC) (Haarnoja et al.2019) is another compelling algorithm, encouraging diverse, exploratory behaviour by maximising entropy; rewarding randomness in actions. In many domains this exploration bonus accelerates learning. However, when attempting to minimise worst-case risk, you often want to approach deterministic, precise hedging strategies. SAC's drive for randomness directly opposes our goal of tight risk control.

In addition, we are able to dispense with the need to create an adversarial agent that learns. Since our Nature is solved by corner enumeration as proved throughout this chapter, we need not require an adversary network to learn a mapping that is already known. This adds additional complexity and introduces stochastici without any upside for our particular problem.

Despite the widespread adoption and viability of PPO, we must adjust it for our purpose. This standard implementation optimises for expected returns, however, our focus on CVaR requires careful modifications to the advantage estimation and policy updates. We must teach the algorithm to care disproportionately about tail events, whilst maintaining the stability benefits of PPO's trust region approach. Essentially, we require special emphasis on worst case scenario, given the risk-based nature of our problem. For this we introduce "RobustCVaR-PPO".

3.7.3 Robust CVaR-PPO

Let us first visit the concept of Actor Critic Reinforcement Learning methods. Actor Crtici algorithms combine both policy based and value based methods. They have an Actor, which as the name suggests, selections actions. A Critic then evaluates them. It is akin to have an external entity providing feedback on your decisions. The formulation is as follows:

1. Actor: follows a Gaussian policy $\pi_\theta(h|s) = \mathcal{N}(\mu_\theta(s), \sigma^2(s)$
2. Critic: a tail value value function $V_\phi = \mathbb{E}[L - \zeta]_+|s]$

where μ is our best guess delta, σ is how nervous we are about that guess and V_ϕ as shown earlier, is the expected loss beyond the current VaR buffer.

As mentioned before, we remove the expensive constraint imposed by TRPO and second derivatives with a more compute-friendly empirically sound and stable clip of

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t}{\pi_{\theta\text{old}}(a_t|s_t}$$

where if r_t drifts outside [1-ϵ, 1 + ϵ] we stop the gradient. We implement the loss functions shown by Schulman et al. (2017) with a slight engineering tweak

$$\boxed{\mathcal{L}_{\text{PPO}}(\theta) \;=\; \mathbb{E}_t\big[\min\!\big(r_t(\theta)\,\hat{A}_t^{\text{tail}},\ \text{clip}(r_t, 1\pm\varepsilon)\,\hat{A}_t^{\text{tail}}\big)\big]}$$

We note the change of $\hat{A}$ to $\hat{A}^{tail}$ which reflects the change in vanilla advantage

$$\hat{A}_t = Q_\theta(s_t, a_t) - V_\phi(S_t)$$

to

$$\hat{A_t}^{tail} = (L_t - \zeta)_+ - V_\phi(S_t)$$

This, however, does not impact our clipping objective and as such we can make use of the compute friendly clipping of PPO ensuring the trust region safety still holds. Revisiting our four constraints, we note how this implementation suits our practical needs

1. Continuous h_t: Gaussian actor outputs μ and σ directly (μ is a real number, so gradient descent can push it to any point on $\mathbb{R}$, therefore search space is continuous

2. Hidden Regimes : Filtered belief state

3. Tail risk: Tail advantage implementation through PPO with ζ tracker

4. Adversary : Cheap four corner enumeration and PPO clipping keeps updates stable even in worst case.

   With the theoretical foundations now in place, we turn to Chapter 4 to the practical methodology and implementation details

4 Methodology

5 Experiments and Results

6 Discussion and Conclusion

7 Bibliography

8 Appendix

8.1 Appendix A: Self financing portfolio intuition

Consider an option trader who has just sold a call option to a client for price C_0. Note that a call option gives a market participant the right, but not the obligation, to purchase the underlying at some specified time in the future. The trader now has a liability. Depending on market movements, the trader could owe the client a payment at expiration. To manage this risk, the trader can create a dynamic hedging portfolio with the aim to offset the option liability. This portfolio consists of two components: some number of shares in the underlying (we denote this as h_t) and some cash invested at the risk-free rate. A constraint that must be satisfied is that it must be a self-financing portfolio. The implications are as follows: the trader cannot inject or withdraw money post-initial setup; any changes to the position must be funded by selling other assets within the portfolio.

Mathematical Framework We denote portfolio value at time t as Π_t consisting of h_t shares worth $h_t \times S_t$ plus some cash position. At any moment total portfolio value is $\Pi_t =$ (value of stock holdings) + (value of cash holdings) The self-financing constraint then appears as follows: suppose the trader decides to change their hedge ratio from h_t to $h_{t+\Delta t}$. If they increase stock holdings, the purchase must be funded by reducing cash holdings by exactly the same amount. Similarly, a reduction of stock holdings results in proceeds added to the cash account.

Rebalancing mechanics At time t, suppose the trader holds h_t shares and some cash. Just prior to time $t + \Delta t$, they observe the new stock price $S_{t+\Delta t}$ and decide to adjust their hedge to $h_{t+\Delta t}$. The change in stock position, multiplied by new price, determines the corresponding adjustment to the cash holdings. However, cash holdings have grown at the risk-free rate r during this time. Therefore, assuming the cash amount X at time t, it becomes $Xe^{r\Delta t}$ at time $t + \Delta t$ when solving its differential equation. We then have

$$X_{t+\Delta t} = Xe^{r\Delta t} - (h_{t+\Delta t} - h_t)S_{t+\Delta t}$$

Portfolio value at time t can be written as the following:

$$\Pi_t = h_t S_t + X_t$$

rearranging:

$$X_t = \Pi_t - h_t S_t$$

after rebalancing at time $t + \Delta t$, the new portfolio value becomes:

$$\Pi_{t+\Delta t} = h_{t+\Delta t} S_{t+\Delta t} + X_{t+\Delta t}$$

substituting the expression for cash holdings

$$\Pi_{t+\Delta t} = h_{t+\Delta t} S_{t+\Delta t} + (\Pi_t - h_t S_t) e^r \Delta t - (h_{t+\Delta t} - h_t) S_{t+\Delta t}$$

simplifying the stock terms by expanding brackets:

$$\Pi_{t+\Delta t} = (\Pi_{t-h_t S_t}) e^{r\Delta t} + h_t S_{t+\Delta t}$$

finally subtracting the call option liability:

$$\Pi_{t+\Delta t} = (\Pi_{t-h_t S_t}) e^{r\Delta t} + h_t S_{t+\Delta t} - C_{t+\Delta t}$$

Economic Implications The self-financing constraint dictates profits or losses within the hedging strategy come purely from the difference between the hedge portfolio performance versus the option liability. Assuming efficient markets and continuous hedging (cornerstones of Black-Scholes assumptions), a trader could hedge continuously with zero transaction costs. This would lead to a perfect replication of the option payoff, making the hedging shortfall zero. However, reality dictates rebalancing occurs discretely, markets have transaction costs, and the underlying model is not perfect. These imperfections create the hedging shortfall that the trader seeks to minimise.

8.2 Appendix B: Option Payoff Introduction

8.3 Appendix C: Overview of reinforcement learning

8.4 Appendix D: Introduction to Value at Risk

VaR is defined as:

$$VaR_a(L) = inf\{l : Pr[L \leq l] \geq \alpha\}$$

(a) L is any random loss we may suffer

(b) α is the confidence interval we care about

(c) $Pr[L \leq l]$ is the probability that the actual loss we see is larger than L

(d) inf is short for infimum i.e. the smallest number in a set

To explain how this works, consider lining up every possible loss threshold on a line. For example, we can have losses of ranging from 0 to 2 million in multiple of 100k. For each of these values, we ask what is the chance our loss is no larger than this value? We can see from the table above, that the infimum i.e. lowest value where the probabililty meets our alpha of 95% is 900k therefore that is our 95% VaR value.

L	$\Pr[L \leq \ell]$	Meets 95%?
500k	94%	×
800k	94%	×
900k	95.3%	✓
1m	96%	✓
2m	98%	✓

8.5 Appendix E: self-adjustment of VaR

8.6 Appendix F: Isses with just using CVaR