Risk Management for Options using Deep Generative Modeling of Risk-Neutral Distributions

Rohit Barve[a,b], Aniket Bhanu[a], Rejaul Barbhuyan[a]

NSE Clearing Limited

Abstract

Risk management models for cleared options at Central Counterparties (CCPs) conventionally rely on simulations of risk factors such as underlying prices and implied volatilities to generate option price returns. These models compress rich market information and introduce model-dependent biases. This paper presents a model-independent methodology for simulating option price returns by capturing changes in risk-neutral distributions (RNDs) derived from historical option prices. The approach involves training a Tabular Variational Autoencoder (TVAE), a deep generative model, to learn and generate daily variations in RND based on observed historical patterns. The approach is evaluated on Indian options market data and demonstrates superior performance to Monte Carlo simulations in capturing option price return distributions. This method provides CCPs with a more accurate framework for risk measurement, margin setting, stress scenario design, and model validation.

Keywords : Central Counterparties (CCPs), risk management, scenario generation, risk-neutral distribution, generative modeling, autoencoders

---

[a] Any views expressed in the paper are solely of the authors and may not represent the views of NSE Clearing Limited. The authors thank Prof. Golaka Nath for his useful comments and suggestions.

[b] Corresponding Author: rbarve@nsccl.co.in

1 Introduction and Related Literature

Central counterparties (CCPs) play a critical role in financial market stability by interposing themselves between trading counter parties and guaranteeing settlement (Domanski, Gambacorta, & Picillo, 2015). To fulfill this mandate, CCPs must maintain a robust risk management framework that includes the setting of initial margins, the construction of stress scenarios, and the validation of margin and risk models. These processes are essential to ensure that potential portfolio losses are covered with a high confidence over a defined margin period of risk, appropriate financial safeguards are built to ensure resiliency in extreme but plausible market conditions, and margin and stress testing methodologies are appropriate, robust and responsive.

Options markets pose unique challenges for these tasks: contracts roll forward, new strikes and expiries are listed, liquidity shifts across the chain, and portfolio P&Ls depend upon joint dynamics across strikes and maturities. CCP risk models typically address them by using risk-factor based approaches which compress complex option surfaces into simplified parametric models. Such models apply shocks to a small set of risk factor returns, typically underlying price and implied volatility (IV), and estimate option price returns using theoretical pricing models based on such risk-factors. Instead, we propose a model-independent approach that represents option surfaces in risk-neutral distribution space, enabling more accurate price simulations through a deep generative model.

1.1 Risk management models for options and their limitations

Risk management models for options typically involve simulating changes in underlying risk factors and using an options pricing model to obtain prices given the changes in these risk factors. This simulation-driven methodology ties the price of an option to the key variables affecting the price such as spot price, volatility, interest rates and captures non-linear sensitivities.

The primary risk factors affecting option prices are the returns and volatility of the underlying asset. Prior studies propose several methods for simulating these risk factor returns. For example, Monte Carlo methods can be used to simulate risk factor returns by generating paths for these risk factors (Glasserman, 2004, Boyle, 1977). Alternatively, Filtered Historical Simulation (FHS) adjusts historical factor returns to align with current market conditions by scaling the volatility of returns (Barone-Adesi, Giannopoulos, & Vosper, 1999). After simulating risk factor returns using an appropriate method, option prices for each simulated state can be calculated using a suitable options pricing model. This may involve the classical Black-Scholes-Merton model (Black & Scholes, 1973) or models that account for specific market features, such as return discontinuities (Merton, 1976), stochastic volatility (Heston, 1993), or variations in implied volatility surfaces (Hagan, Kumar, Lesniewski, & Woodward, 2002).

This simulation-based approach is often necessary due to the structure of options markets. Option contracts typically have short lifespans, with liquidity concentrated in specific periods. As new expiries are introduced and strike sets adjust to changes in the underlying asset's price, liquidity tends to focus on certain strikes and expiries (Etling & Miller Jr, 2000). As a result, option chains for liquid options over extended periods are not directly comparable. To address these challenges of rolling maturities and sparse market quotes, implied volatilities are frequently organized into

delta/moneyness and tenor grids (Eurex Clearing, 2022), or represented using parametric smile models. This consolidates the cross-sectional information from the full surface of strikes and maturities into a reduced set of risk factors, losing some price information in the process.

Implied volatility is derived from market prices through a process involving price inversion using a selected pricing model. Smiles are fitted using parameterized models or interpolation schemes. Theses steps introduce modeling assumptions that may bias the resulting scenarios. Additionally, pricing models are sensitive to parameter choices, which can affect the accuracy of the simulations (Christoffersen & Jacobs, 2004).

Deep learning models can improve prediction accuracy of financial derivatives pricing since financial markets exhibit stochastic volatility and jumps (Jang & Lee, 2019). Generative Adversarial Networks (GANs) have been found to outperform conventional Monte-Carlo based simulations for option pricing (Choi, Ryu, Byun, Na, & Song, 2025), portfolio hedging (Buehler, Gonon, Teichmann, & Wood, 2019), etc. However, we did not come across any literature that applies deep generative models to simulate option price returns for risk management. We propose extending generative modeling techniques to construct realistic return distributions for options, thereby enabling more robust risk assessment and stress testing frameworks.

1.2 Proposed model and contribution

We propose an alternative framework that uses all available price information without imposing buckets for implied volatility. We estimate the risk-neutral cumulative distribution function $F_T^Q(x) = \mathbb{Q}(S_T \leq x)$ from observed option prices. We represent the result on a standardized moneyness grid so that surfaces are comparable as contracts roll. $F_T^Q(x)$ denotes the cumulative probability distribution of the underlying price at a time T, which when used to compute the expected payoff of an option (conditional on it being in the money) and discounted at the risk-free rate, reproduces the observed market price of the option. Instead of simulating future values of risk factors and obtaining option prices with a pricing model, we simulate day-over-day changes $\Delta F_T^Q(x)$, and then apply such changes on the current $F_T^Q(x)$ to simulate it's future values. From such simulated set of $F_T^Q(x)$, we obtain option prices by discounting the expected payoff under such distributions.

To model and generate synthetic $\Delta F_T^Q(x)$ values and simulated option prices, we adopt the following approach:

1. We obtain $F_T^Q(x)$ from observed option prices and estimate their values over a standard moneyness grid $\mathcal{M}$. It is emphasized that $F_T^Q(x)$ is constructed using all available option price information. Since such estimation of $F_T^Q(x)$ is non-parametric, day-over-day changes in its values can only be constructed by sampling its values for a large sized standardized grid.

2. We calculate daily changes $\Delta F_T^Q(x) \ \ \forall x \in \mathcal{M}$.

3. We apply principal component analysis (PCA) to obtain a set of components Z that explain most of the variation in $\Delta F_T^Q(x)$.

4. We train a Tabular Variational Autoencoder (TVAE) model to learn the conditional distribution of Z using time-to-expiry τ as the conditioning variable.

5. We generate simulations of Z using the trained TVAE model conditioned on τ.

6. We obtain $\Delta F_T^Q(x)$ from simulations of Z by applying inverse PCA.

7. For a given day t, we simulate next day's $F_T^Q(x)$ as:

$$F_{T,t+1}^Q(x) = F_{T,t}^Q(x) + \Delta F_T^Q(x)$$

8. We obtain prices for different option contracts using $F_{T,t+1}^Q(x)$.

We use data from the Indian options market, one of the largest in the world in terms of number of contracts traded to test the proposed methodology. We train the TVAE model on four years of training data and test it on one year of out-of-sample data, focusing on two maturities (weekly and monthly) with three buckets of moneyness per day. We evaluate the proposed method against a benchmark Monte Carlo simulation which simulates shocks to the underlying asset price and volatility, and then reprices options using these simulated shocked values. Using the continuous ranked probability score (CRPS), we measure how well model implied price change distributions matches the actual option price returns. We find that the proposed method consistently performs better than Monte Carlo simulations.

Our proposed methodology may be valuable for risk management in CCPs including margining, stress test design, model validation etc. This methodology provides distribution of option price returns, which can be used directly for risk measures like Value-at-Risk (VaR) and Expected Shortfall (ES), setting margin requirements, designing stress scenarios, and validating models. It addresses the operational limitations discussed previously: (i) it restores comparability under rolling contracts by mapping risk-neutral distribution onto a standardized moneyness grid, which turns the daily, irregular option chain into a fixed-dimension representation. This eliminates the need for IV buckets or parametric surface models. (ii) it uses the full set of observed option prices rather than compressing the surface into a handful of buckets. (iii) it avoids model-dependent smile assumptions by working directly with the distribution implied by prices, without inverting to implied volatilities. In this way, the approach remains model-independent at the scenario level.

The remainder of the paper is structured as follows: Section 2 gives an introduction to risk neutral distributions and deep generative modeling. Section 3 describes the methodology in detail. Section 4 presents the empirical analysis and results. Section 5 concludes the study.

2 Background on Risk Neutral Distributions and Deep Generative Models

2.1 Risk Neutral Distributions

The risk-neutral distribution is the distribution of the underlying asset's price at expiration such that, when used to compute the expected payoff of an option (conditional on it being in the money) and discounted at the risk-free rate, it reproduces the observed market price of the option. In other words, if price of a call option at strike price K and expiry T (and remaining time to expiry $\tau = T - t$) is $C_{K,T}$ and the risk -neutral probability density function is $f_T^Q(x)$, then

$$C_{K,T} = e^{-r\tau} \int_{S_T=K}^{\infty} (S_T - K)\ f_T^Q(S_T)\ dS_T \tag{1}$$

Differentiating twice with respect to K and rearranging, we get

$$f_T^Q(K) = e^{r\tau}\ \frac{\partial^2 C_{K,T}}{\partial K^2} \tag{2}$$

This result is known as the Breeden-Litzenberger formula (Breeden & Litzenberger, 1978).

In our paper, we use the cumulative distribution function $F_T^Q(K)$ instead of the density function $f_T^Q(K)$ because the monotonicity and [0,1] bounds of the $F_T^Q(K)$ make it easier to model with neural networks than the density function $f_T^Q(K)$. The expression for $F_T^Q(K)$ can be obtained by taking the first derivative of equation 1,

$$\frac{\partial C_{K,T}}{\partial K} = -e^{-r\tau} \int_{S_T=K}^{\infty} f_T^Q(S_T)\ dS_T \tag{3}$$

then [1],

$$\int_{S_T=K}^{\infty} f_T^Q(S_T)\ dS_T = 1 - F_T^Q(K) \tag{4}$$

Thus, the cumulative distribution function $F_T^Q(K)$ will be,

$$F_T^Q(K) = 1 + e^{r\tau}\ \frac{\partial C_{K,T}}{\partial K} \tag{5}$$

[1] Since $F_T^Q(K) = \int_{-\infty}^{K} f_T^Q(S_T)\ dS_T$ and $\int_{-\infty}^{K} f_T^Q(S_T)\ dS_T + \int_{K}^{\infty} f_T^Q(S_T)\ dS_T = 1$

In practice, this derivative can be computed using finite differences over adjacent strikes as,

$$F_T^Q(K_i) \approx 1 + e^{r\tau} \frac{C(K_{i+1}, T) - C(K_i, T)}{K_{i+1} - K_i} \tag{6}$$

Also, if discrete points of $F_T^Q(K_i)$ are available $C_{K,T}$ can be calculated as a discrete equivalent of equation 1,

$$C_{K,T} \approx e^{-r\tau} \sum_{i=K}^{\infty} \frac{1 - F_T^Q(K_i) + 1 - F_T^Q(K_{i+1})}{2} \Delta K \tag{7}$$

2.2 Deep Generative Models

2.2.1 Background

Generative models are a type of machine learning models designed to generate new, synthetic data such that it closely resembles the data they were trained on (Jebara, 2012). Unlike discriminative models, which focus on mapping the inputs to outputs, generative models aim to learn the underlying data distribution so that they can create realistic samples mimicking the given dataset from this learned distribution. This property is particularly useful in finance, where realistic synthetic datasets can support tasks such as scenario generation, stress testing, and data augmentation in settings where historical records are limited.

Deep generative models are a class of generative models which rely on neural networks to learn the underlying structure of data and to generate new observations that resemble the original. Unlike traditional models, which assume an explicit data distribution and learn the parameters of this distribution, in generative deep models, the weights of the neural network are trained such that they can recreate the original data and the parameters are learned implicitly.

2.2.2 Variational Autoencoders

Autoencoders are a type of deep learning models which map data into a lower-dimensional latent space representation and then reconstruct the data from this representation (Hinton & Salakhutdinov, 2006). Conceptually, an autoencoder consists of two complementary components: an encoder and a decoder. The encoder compresses the input data into a lower-dimensional, compressed latent representation while the decoder reconstructs the original input from this compressed representation. By learning to reproduce its input as accurately as possible, the autoencoder identifies the essential patterns and structure present in the dataset. Essentially, the latent space learned by the autoencoder acts as a parameterization of the data distribution.

Let $x \in \mathbb{R}^d$ denote a single data point in the high-dimensional space. The encoder is defined as a parametric mapping

$$f_\phi : \mathbb{R}^d \to \mathbb{R}^k$$

where k is the dimensionality of the latent space such that $k \ll d$ and ϕ represents the encoder parameters. These parameters are essentially the weights of the underlying neural network that is trained as part of the autoencoder. If the latent representation is denoted by z, then it can be represented as

$$z = f_\phi(x), \quad z \in \mathbb{R}^k$$

Similarly, the decoder can be defined as the mapping

$$g_\theta : \mathbb{R}^k \to \mathbb{R}^d,$$

which aims to reconstruct the original input from the latent representation z. Here, θ denotes the decoder parameters. The output of the decoder can be denoted as

$$\hat{x} = g_\theta(f_\phi(x))$$

The training objective is to find parameter values (ϕ, θ) that minimize the discrepancy between the input x and its reconstruction $\hat{x}$. This discrepancy is often quantified using a reconstruction loss, $\mathcal{L}$, given by:

$$\mathcal{L}(\phi, \theta) = \frac{1}{N} \sum_{i=1}^{N} \left\| x^i - g_\theta(f_\phi(x^i)) \right\|^2$$

where N is the number of training samples.

Gradient descent algorithm is used to minimize the reconstruction loss in Autoencoders by iteratively adjusting the model's parameters (weights of the neural network) to reduce the difference between the input and its reconstructed output. It works by computing the gradient of the loss with respect to the parameters via backpropagation, then updating them in the direction that lowers the loss, typically using small steps scaled by a learning rate. However, as Autoencoders represent the input data as a single deterministic latent vector, z, they are unable to generate new, synthetic data points resembling the original data. Sampling random z values would yield random and irrelevant outputs. These models are optimized for dimensionality reduction and reconstruction of original data from the latent vector.

Variational Autoencoders (VAEs) are a class of Autoencoders that learn the parameters of a probabilistic distribution over the latent space, rather than a single deterministic latent vector, enabling generative modeling by imposing a structured, continuous latent space (Kingma & Welling, 2013). Unlike the single latent vector in Autoencoders, this provides a smooth and continuous latent space which is suitable for sampling.

Let $x \in \mathbb{R}^d$ denote a single data point in the high-dimensional space and $z \in \mathbb{R}^k$ represent the lower-dimensional latent variable where $k \ll d$. In a Variational Autoencoder, the encoder will map the input x to a probability distribution $q_\phi(z|x)$ and learn the parameters ϕ of this distribution. Typically, a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ is used in the probabilistic latent space and the encoder learns the parameters μ and σ^2 of this distribution.

A decoder is a neural network which takes in the latent variable z from the encoder output and will output the distribution $p_\theta(x|z)$, where θ represents the parameters of decoder. It will then generate new data $\hat{x} \sim p_\theta(x|z)$ from this distribution. During the training run, a loss function follows the decoder to calculate the difference between the original data and the generated data. However, in a VAE due to the inherent randomness in the latent space, the reconstructed data $\hat{x}$ will

differ from the original data x unlike the Autoencoder model where the original and reconstructed data are supposed to match as closely as possible. Therefore, in a VAE, the loss function measures how well the distribution $p_\theta(x|z)$ can produce outputs close to the real data x.

The loss function in the VAE is the negative Evidence Lower Bound (ELBO) and it aims to maximize this value. It tries to approximate the likelihood $\log p_\theta(x|z)$ of the data. The Evidence Lower Bound is defined as:

$$\mathcal{L}_{ELBO} = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)\|p(z))$$

Tabular Variational Autoencoders (TVAE) are an extension of VAE which are designed to handle continuous and categorical variable data in a tabular format (Xu, Skoularidou, Cuesta-Infante, & Veeramachaneni, 2019). VAEs are designed for smooth and continuous domain data such as images but struggle with data with mixed data types. To accommodate tabular data with mixed data types, TVAEs make a few modifications: (i) they use mode-specific normalization to pre-process the rows. Continuous columns are split into a scalar (α), which is the normalized value within a mode and one-hot vector (β), the mode from a fitted Gaussian mixture. Discrete columns (d) are kept as one-hot encoded vectors. (ii) TVAE models a joint distribution for the mixed types - Gaussian distribution for the scalar (α) and softmax for the mode indicator (β) and discrete value (d). (iii) TVAE modifies the ELBO loss to account for these joint distribution values.

2.2.3 Generative Adversarial Networks

Generative Adversarial Networks (GANs) consist of two neural networks trained in opposition: a generator G and a discriminator D (Goodfellow et al., 2014). The generator maps a latent variable $z \sim p(z)$, typically drawn from a simple prior distribution such as a multivariate Gaussian or uniform distribution, into the data space, producing synthetic examples $\hat{x} = G_\theta(z)$. The discriminator receives either real samples $x \sim p_{\text{data}}$ or generated samples $\hat{x}$, and outputs a probability $D_\phi(x) \in [0, 1]$ representing probability of the input being a real sample (rather than being a generated sample). Training is formulated as a two-player minimax game:

$$\min_\theta \max_\phi \; \mathbb{E}_{x\sim p_{\text{data}}}\big[\log D_\phi(x)\big] + \mathbb{E}_{z\sim p(z)}\big[\log(1 - D_\phi(G_\theta(z)))\big].$$

The discriminator is trained to maximize its ability to distinguish between real and generated data, while the generator is trained to minimize this objective by producing samples that fool the discriminator. At the theoretical Nash equilibrium of this game, the generator distribution p_G matches the true data distribution p_{data}, and the discriminator outputs $D_\phi(x) = \frac{1}{2}$ for all x, meaning it can no longer distinguish real from synthetic samples. In practice, D_ϕ and G_θ are updated alternately: the discriminator is first trained to improve classification accuracy, then the generator is updated using gradients propagated through the discriminator to improve sample quality.

Despite their theoretical appeal and success in domains such as image synthesis, GANs are difficult to train reliably. In our study, we found the GAN to suffer from mode collapse (Goodfellow et al., 2014, Kossale, Airaj, and Darouichi, 2022). Mode collapse occurs because the generator discovers and exploits a few modes that successfully fool the discriminator while ignoring other

regions in the data distribution [2]. For this reason, despite their theoretical potential, GANs proved impractical for our objectives.

3 Proposed Methodology

This section provides the details of the proposed methodological framework. Figure 1 provides a diagrammatic representation of the methodology, which is discussed in detail in the subsequent section.

[Figure 1 about here]

3.1 Pre-processing

We are given an observed option chain of call and put options over a range of strike prices. Let $C_{K,T}$ and $P_{K,T}$ represent the call and put price of the option at a strike K and expiry T. We correct illiquidity-induced mispricing in the option chain with a two-step approach (Cohen, Reisinger, & Wang, 2020):

1. **Initial price adjustment to ensure Put-Call parity**

   To correct for imbalances in liquidity between call and put options, the put-call parity principle (Hull, 2009) is applied. For strikes where the volume of put contracts exceeds that of calls, the call price is corrected using the corresponding put price, and vice versa. The adjusted prices are

   $$\begin{aligned} C_{K,T} &\longleftarrow P_{K,T} + S_0 - Ke^{-r\tau}, \quad \text{if } V_{P_{K,T}} > V_{C_{K,T}} \\ P_{K,T} &\longleftarrow C_{K,T} - S_0 + Ke^{-r\tau}, \quad \text{if } V_{C_{K,T}} > V_{P_{K,T}} \end{aligned}$$

   where $V_{C_{K,T}}$ & $V_{P_{K,T}}$ represent the trading volumes of call & put options respectively.

   After ensuring Put-Call parity, only the call prices are used for subsequent analysis.

2. **Other Adjustments to ensure Arbitrage-Free Prices**

   The option prices are constrained by the following additional conditions:

   (a) Monotonicity of Call Prices: $C_{K_i,T} \geq C_{K_{i+1},T}$

   (b) Strike Sensitivity Constraint: $\left|C_{K_{i+1},T} - C_{K_i,T}\right| \leq |K_{i+1} - K_i|$

   (c) Butterfly Arbitrage Constraint (Non-negativity):

   $$C_{K_{i-1},T} - 2\,C_{K_i,T} + C_{K_{i+1},T} \geq 0$$

[2] Results of GAN implementation on our data which was found to lead to mode collapse can be provided on request.

The optimization problem subject to the above constraints can be formalized as:

$$\min_{C_{K_i,T}} \sum_i \left(C^{obs}_{K_i,T} - C_{K_i,T}\right)^2$$

where $C^{obs}_{K_i,T}$ is observed market price of the call option $C_{K_i,T}$.

3.2 Estimation of risk neutral distribution

Having obtained arbitrage-free call option prices, the values of cumulative risk-neutral distribution $F^Q_T(x)$ can be obtained at each of the strikes K_i using equation 6.

Since option chains exhibit different ranges of available strikes across maturities due to changes in the underlying asset's price, we transform the underlying price into moneyness (K_i/S) by dividing the strike price K_i with the contemporaneous spot price S. Next, we desire to estimate the values of $F^Q_T(x)$ on a standardized grid of moneyness $\mathcal{M}$ for each of the maturities T to enable identification of differences in $F^Q_T(x)$ across days. We train a separate neural network for each trading day and each expiry T to learn the mapping from moneyness to the $F^Q_T(x)$.

$$g_T : \mathbb{R} \to [0,1], \quad g_T(x) \approx F^Q_T(x)$$

where $x \in \mathbb{R}$ denotes the moneyness value. Once trained, we evaluate the mapping on the standard grid of moneyness values $\mathcal{M}$ obtaining,

$$F^Q_T(m) \approx g_T(m), \quad m \in \mathcal{M}$$

3.3 Feature creation

For each trading day t, difference in the values of $F^Q_T(m)$ over successive days for each expiry T is calculated as

$$\Delta F^Q_{T|t}(m) = F^Q_{T|t+1}(m) - F^Q_{T|t}(m) \tag{8}$$

When the moneyness grid $\mathcal{M}$ is finely defined, the number of $\Delta F^Q_T(m)$ values becomes very large, which introduces computational difficulties in training the generative model. Many of the values in the grid, especially those deep in and out of the money, may also be invariant. Therefore, we carry out Principal Component Analysis (PCA) to reduce the dimensionality of the $\Delta F^Q_T(m)$, which yields principle components Z_k which explain most of the variance in the data.

3.4 Model training

We construct a training dataset of the principal components Z_k and time to expiry $\tau = T - t$, for trading date t and expiry T. We train a Tabular Variational Autoencoder (TVAE) model on this dataset to sample synthetic Z_k values conditioned on τ. For the purpose of this study, we used the implementation of the TVAESynthesizer model from the `sdv` package (Xu et al., 2019).

3.5 Generation of simulated option prices

Suppose it is desired to generate returns of option prices for an out-of-sample day d. We generate simulations of Z_k for a given $\tau = T - d$ using the trained TVAE model. We then generate corresponding values of $\Delta F^Q_{T,d}(m)$ by applying inverse PCA on Z_k simulations. We generate simulations for next day's cumulative risk-neutral distribution as

$$F^Q_{T,d+1}(m) = F^Q_{T,d}(m) + \Delta F^Q_{T,d}(m) \tag{9}$$

The resulting synthetic $F^Q_{T,d+1}(m)$ set of values spans over the moneyness grid $\mathcal{M}$ defined earlier. We obtain the spot price S corresponding to each of the moneyness values $m = K/S$ by multiplying them with the underlying price on day d.

Finally, we can then calculate the call option prices using equation 7.

3.6 Benchmarking

3.6.1 Monte-Carlo Simulation

In this paper, we use Monte-Carlo simulation to benchmark the effectiveness of the proposed framework. We simulate the returns in the underlying price and implied volatility, and then use these values in the Black-Scholes pricing model to calculate option price returns under these simulations (Alexander, 2009). The summary of the methodology is as follows:

1. Risk factor specification: We consider two primary risk factors that drive option prices over the training period: the daily log returns of the underlying asset price and the implied volatility. In practice, CCPs use buckets based on moneyness and tenor for modeling implied volatility returns (Eurex Clearing, 2022). Accordingly, we have used such bucketing.

2. Generation of simulations: We construct the covariance matrix of the risk factors and using it's Cholesky decomposition, generate simulations having the joint distribution of returns in accordance with the covariance matrix.

3. Generation of option prices: We use these simulated risk factors as an input to the Black-Scholes pricing model to generate simulated option prices and their log returns.

3.6.2 Continuous Ranked Probability Score (CRPS)

We aim to benchmark the option price return simulations produced by the deep generative model against those generated through Monte Carlo simulation. As the deep generative model yields a non-parametric distribution without a closed-form representation, conventional evaluation methods such as log-likelihood cannot be applied. Therefore, we employ the CRPS as the evaluation metric (Matheson & Winkler, 1976).

The CRPS is a proper scoring rule used to evaluate the quality of distributional forecasts by comparing a predictive distribution with the observed outcome. Let the predictive distribution be defined on a finite or countable set $\{x_1, x_2, \ldots, x_n\}$ with cumulative distribution function (CDF) $F(x_k) = \sum_{i=1}^{k} p_i$, where p_i denotes the probability assigned to outcome x_i. If the realized outcome is y, the CRPS for the discrete case can be written as

$$\mathrm{CRPS}(F, y) = \sum_{k=1}^{n} \left(F(x_k) - \mathbf{1}_{\{x_k \geq y\}}\right)^2 \Delta x_k,$$

where Δx_k represents the step size between consecutive support points (often taken as one when the support is equally spaced), and $\mathbf{1}_{\{x_k \geq y\}}$ is the indicator function that equals one if $x_k \geq y$ and zero otherwise. This summation form is the discrete analogue of the integral definition in the continuous setting. Lower CRPS value indicates a better performance of a distributional forecast.

4 Empirical Analysis and Results

National Stock Exchange of India (NSE) is one of the leading markets in the world offering multiple asset classes. We use the options market data from NSE's equity derivatives market, which is one of the largest derivatives market in the world in terms of number of contracts traded. We consider the data for options based on the NIFTY index - which is the flagship stock market index for NSE. We use NIFTY index options over a four-year training period (March 2019 - December 2023) with a one-year out-of-sample test data (January 2024 - December 2024). We use a four-year training period because it yields about 1000 historical days for simulation which matches the historical look-back period used by CCP margin models in practice [3]. In the training and test data, we considered weekly and monthly expiries of NIFTY.

4.1 Analysis

4.1.1 TVAE: Model Training and Simulations

To train the model, the risk-neutral cumulative distribution functions $F_T^Q(x)$ were estimated for each trade date and expiry (weekly and monthly) during the training period. Considering the available market prices of traded options, the value of $F_T^Q(x)$ was estimated at available strikes. A deep neural network is fitted to estimate the values of the $F_T^Q(m)$, for $m \in \mathcal{M}$ over a standardized grid $\mathcal{M}$ [4]. The standardized grid $\mathcal{M}$ spans 100 equidistant moneyness values over the range [0.9, 1.1].

Day-over-day changes in $\Delta F_T^Q(m)$ are calculated and Principal Components explaining most of the variance in $\Delta F_T^Q(x)$ are identified using PCA. It was observed that 5 components explain 97.6% of the variance in $\Delta F_T^Q(m)$. The 5 components were selected for further analysis.

[3] For example, Eurex Clearing uses 750 historical observations (Eurex Clearing, 2022), JSCC uses 1250 historical observation days (Japan Securities Clearing Corporation, 2025).

[4] The network has 4 hidden layers with ReLU activation function (Glorot, Bordes, & Bengio, 2011).

After training the TVAE, the trained model was used to generate 1000 simulations of $\Delta F_T^Q(m)$ [5]. The risk neutral distribution was calculated for each trade date and expiry for the out-of sample period and then option prices and returns corresponding to each simulation of $\Delta F_T^Q(m)$ were calculated.

4.1.2 Monte-Carlo: Feature Selection and Simulations

For each trading day, three moneyness buckets (K/S) were defined as follows:

In-the-money (ITM): (0.97, 0.99),
At-the-money (ATM): (0.99, 1.01),
Out-of-the-money (OTM): (1.01, 1.03)

These buckets were created for weekly and monthly expiries separately, giving a total of six buckets. The most liquid contract for each of the buckets for a given trading day was considered and the return of implied volatility for such representative contract over the day was considered for that bucket. Using the covariance matrix of the risk factors (i.e. underlying price and implied volatility), 1000 simulations were generated. For each of the trading day during the test period, the risk factors were shocked by the generated simulations and the option prices and returns were calculated using Black-Scholes model.

4.1.3 Benchmarking

We considered all liquid contracts in the test period for evaluation. Distribution of option price returns from Monte-Carlo simulation and TVAE model were considered. CRPS values for calculated for each contract. For the purpose of benchmarking, we aggregated the CRPS values for each bucket to get 6 CRPS values for each trading date.

4.2 Results

Figure 2 shows the risk-neutral density and corresponding cumulative distribution function $F_T^Q(x)$ for a sample date 2019-11-25 and weekly expiry ($\tau = 3$). It is observed that the density function is not smooth and has spikes at certain strikes. The phenomenon of "pinning", where underlying asset closes near certain strikes has been empirically observed and can explain such spikes in the density function (Avellaneda and Lipkin, 2003, Golez and Jackwerth, 2012). The corresponding $F_T^Q(x)$ is obtained in accordance with equation 6. The figure shows the $F_T^Q(x)$ estimated by the neural network trained on the observed values.

[Figure 2 about here]

[5] This is in line with approximate 1000 simulations used for historical VaR in CCPs.

Such $F_T^Q(x)$ are estimated for each trading date and expiry. Figure 3 shows the values of $F_T^Q(x)$ for a given expiry (2019-11-28) for two separate trading dates (2019-11-25 and 2019-11-07). The plotted values correspond to neural network estimates on a standardized grid of moneyness (K/S). As expected, the F_T^Q plots become steeper as the τ reduces.

[Figure 3 about here]

The day-over-day changes $\Delta F_T^Q(x)$ values for a given time to expiry τ are constructed from the training data. Figure 4 shows the values of such changes for $\tau = 3$ from the training data. It can be observed that there is a lot of variability in the $\Delta F_T^Q(x)$ values in the ATM region while ITM and OTM regions show less variability. The $\Delta F_T^Q(x)$ values indicates the rate of change in the cumulative distribution. We observe that training data encompasses a diverse set of scenarios with sharp as well smooth changes in the risk-neutral distribution.

[Figure 4 about here]

The historical observations of the changes $\Delta F_T^Q(x)$ conditioned on time to expiry $\tau = T - d$ are applied to a risk-neutral cumulative distribution $F_{T,d}^Q(x)$ to obtain simulated $F_{T,d+1}^Q(x)$ values. Figure 5 shows the simulated $F_{T,d+1}^Q$ (in grey) and the average of such $F_{T,d+1}^Q$ (in red) for sample date 2019-11-25. The simulated $F_{T,d+1}^Q$ exhibits more variation in the ATM region as the probability density function is more concentrated around the underlying price. In contrast, the $F_{T,d+1}^Q$ shows less variation in the ITM and OTM regions as they are present in the tail-ends of the distribution and probability density values being very low.

[Figure 5 about here]

A distribution of forecasted log returns are obtained from the both option prices generated with Monte-Carlo simulation and the TVAE model for each trade date and expiry corresponding to a strike. Figure 6 shows the distribution of returns for an ATM strike generated with the Monte-Carlo simulation and TVAE model along with the actual return value. The TVAE model produces returns closer to the actual return as seen for the sample date 2024-12-05.

[Figure 6 about here]

For a given trade date and expiry in the test set, we calculate CRPS for each available strike in the moneyness and tenor bucket. We then aggregate the results such that we get a single CRPS value for each bucket for each trade date. Table 1 shows summary of key statistics of the CRPS values for the defined buckets. It is observed that the TVAE model outperforms the Monte-Carlo simulation on average (0.586 vs 0.943). It can be seen that TVAE outperforms Monte Carlo on five out of the six buckets, except for OTM weekly contracts. The CRPS values for TVAE for OTM weekly contracts also have a presence of outliers. It can be seen that TVAE outperforms Monte Carlo on median (1.084 vs 1.69) and third quartile values (2.074 vs 2.232) , despite faring worse on average (1.894 vs 1.709).

[Table 1 about here]

In figure 7, we compare the distribution of CRPS values for each of the defined buckets. It is be noted that the distribution closer to zero (lower CRPS values) shows a better performance. The TVAE model shows outperforms the Monte-Carlo simulation method for ITM and ATM contracts along both the maturities. The score distributions show significant overlap for OTM contracts. OTM region in a risk-neutral cumulative distribution has little variability for the TVAE to capture and generate a diverse set of scenarios.

[Figure 7 about here]

We check if the CRPS values show a temporal variability in the performance of the TVAE model as compared to the Monte-Carlo simulation. For each bucket in the given trade date, we calculate the difference in the CRPS values for each contract. We scale these values as

$$C_{scaled} = \frac{C}{max(|C_{max}|, |C_{min}|)}$$

where C_{max} is the greatest positive CRPS value and C_{min} is the least negative CRPS value among all contracts belonging to a given bucket for a given trade date.. Figure 8 shows the performance of the TVAE model for all the buckets along the trade dates in the test set. Positive values (shown in green) indicate a better performance by the TVAE model while negative values (shown in red) indicates that Monte-Carlo simulation was better. The positive values range from light green (0 to 0.3), green (0.3 to 0.6) and dark green (0.6 to 1.0) and the negative values range from light red (0 to -0.3), red (-0.3 to -0.6) and dark red (-0.6 to -1.0). We observe that TVAE performance is better than Monte Carlo - across buckets and trade dates. There is no evidence to suggest that there are regimes when Monte Carlo outperforms TVAE.

[Figure 8 about here]

5 Conclusion

Modeling of option contract returns is an important task for the purposes of risk management. Traditionally, approaches have relied on applying shocks to underlying risk factors and generating simulated option prices based on these risk factors. Furthermore, these approaches are constrained by model-specific assumptions and often compress the option price surface into a set of coarse buckets. Our proposed approach directly simulates the changes in the risk-neutral cumulative distribution and generates a distribution of option price returns from these simulations. Additionally, our approach is pricing model-independent, handles the rolling nature of option chains with irregular set of strike prices, and utilizes the full set of observed option prices without any bucketing or volatility surface representations. The results demonstrate that the proposed method consistently outperforms Monte-Carlo simulations for generation of simulated option price returns.

While the current approach demonstrates strong performance against the benchmark method, there remains further scope for enhancement. The proposed approach can be extended into a filtered historical simulation–styled model, with the generated values conditioned on current day's

RND. Such conditioning shall allow the model to capture prevailing market conditions and output synthetic RNDs which reflect the current regime. We employed PCA to reduce dimensionality of the data for the purposes of computational efficiency. Model training without PCA may improve results further at the cost of higher computational requirements. Nonetheless, even with these simplifications, the proposed framework consistently delivers better results than conventional simulation-based techniques.

Our approach will be particularly valuable for CCPs for generation of option price returns. The output of the model i.e. distribution of returns can be used directly for risk measures such as Value-at-Risk (VaR) and Expected Shortfall (ES) as well as for setting margin requirements. Additionally, these approach can be used for supplementing the stress scenario design process and conducting model validations.

References

Alexander, C. (2009). *Market risk analysis, value at risk models*. John Wiley & Sons.

Avellaneda, M., & Lipkin, M. D. (2003). Amarket-induced mechanism for stock pinning. *Quantitative Finance*, *3*(6), 417.

Barone-Adesi, G., Giannopoulos, K., & Vosper, L. (1999). Var without correlations for portfolios of derivative securities. *Journal of Futures Markets*, *19*(5), 583–602.

Black, F., & Scholes, M. (1973). The pricing of options and corporate liabilities. *Journal of political economy*, *81*(3), 637–654.

Boyle, P. P. (1977). Options: A monte carlo approach. *Journal of financial economics*, *4*(3), 323–338.

Breeden, D. T., & Litzenberger, R. H. (1978). Prices of state-contingent claims implicit in option prices. *Journal of business*, 621–651.

Buehler, H., Gonon, L., Teichmann, J., & Wood, B. (2019). Deep hedging. *Quantitative Finance*, *19*(8), 1271–1291.

Choi, Y. H., Ryu, D., Byun, J. Y., Na, Y., & Song, J. W. (2025). Risk-neutral option pricing via generative adversarial network. *Available at SSRN 5382091*.

Christoffersen, P., & Jacobs, K. (2004). The importance of the loss function in option valuation. *Journal of Financial Economics*, *72*(2), 291–318.

Cohen, S. N., Reisinger, C., & Wang, S. (2020). Detecting and repairing arbitrage in traded option prices. *Applied Mathematical Finance*, *27*(5), 345–373.

Domanski, D., Gambacorta, L., & Picillo, C. (2015). Central clearing: Trends and current issues. *BIS Quarterly Review December*.

Etling, C., & Miller Jr, T. W. (2000). The relationship between index option moneyness and relative liquidity. *Journal of Futures Markets: Futures, Options, and Other Derivative Products*, *20*(10), 971–987.

Eurex Clearing. (2022). Eurex Clearing Prisma - Portfolio-based Risk management. Retrieved from https://www.eurex.com/resource/blob/32818/179aa114c0a294159c37e54f70eab745/data/brochure_eurex-clearing-prisma.pdf

Glasserman, P. (2004). *Monte carlo methods in financial engineering*. Springer.

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (pp. 315–323). JMLR Workshop and Conference Proceedings.

Golez, B., & Jackwerth, J. C. (2012). Pinning in the s&p 500 futures. *Journal of Financial Economics*, *106*(3), 566–585.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, *27*.

Hagan, P. S., Kumar, D., Lesniewski, A. S., & Woodward, D. E. (2002). Managing smile risk. *The Best of Wilmott*, *1*, 249–296.

Heston, S. L. (1993). A closed-form solution for options with stochastic volatility with applications to bond and currency options. *The review of financial studies*, *6*(2), 327–343.

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, *313*(5786), 504–507.

Hull, J. (2009). *Options, futures and other derivatives/john c. hull*. Upper Saddle River, NJ: Prentice Hall.

Jang, H., & Lee, J. (2019). Generative bayesian neural network model for risk-neutral pricing of american index options. *Quantitative Finance*, *19*(4), 587–603.

Japan Securities Clearing Corporation. (2025). Outlines of Margin Calculation Method (VaR Method) for Futures/Option Contracts. Retrieved from https://www.jpx.co.jp/jscc/en/cash/futures/marginsystem/gjnthi000000049n-att/Outlines_VaRMethod.pdf

Jebara, T. (2012). *Machine learning: Discriminative and generative*. Springer Science Business Media.

Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kossale, Y., Airaj, M., & Darouichi, A. (2022). Mode collapse in generative adversarial networks: An overview. In *2022 8th international conference on optimization and applications (icoa)* (pp. 1–6). IEEE.

Matheson, J. E., & Winkler, R. L. (1976). Scoring rules for continuous probability distributions. *Management science*, *22*(10), 1087–1096.

Merton, R. C. (1976). Option pricing when underlying stock returns are discontinuous. *Journal of financial economics*, *3*(1-2), 125–144.

Xu, L., Skoularidou, M., Cuesta-Infante, A., & Veeramachaneni, K. (2019). Modeling tabular data using conditional gan. *Advances in neural information processing systems*, *32*.

Figure 1: Proposed Methodology flowchart

The below figure shows the high-level overview of the proposed methodology

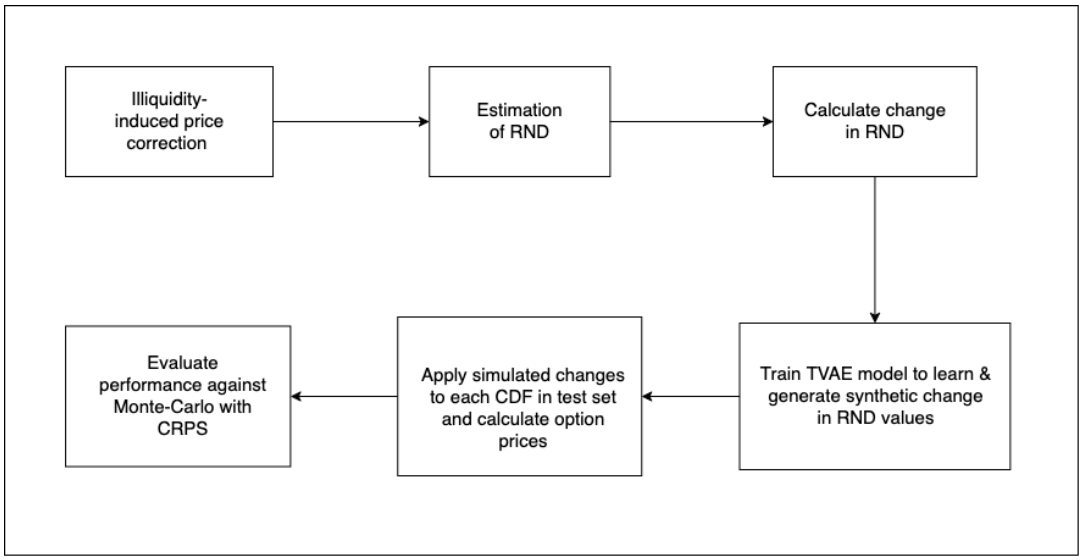


Figure 2: Risk-neutral Density and Cumulative Distribution Functions

The figure shows the risk-neutral density (PDF) and cumulative distribution functions (CDF) for a sample day 2019-11-25 in the training period.

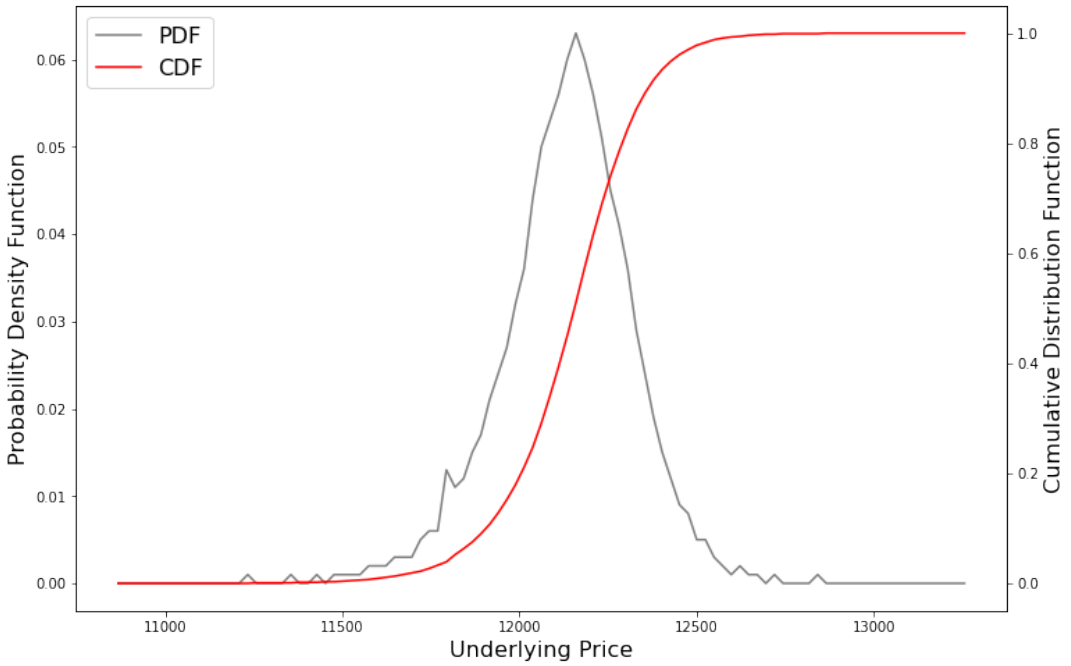


Figure 3: Cumulative Distribution Functions at $\tau = 3$ and $\tau = 21$

The figure shows the cumulative distribution functions for $\tau = 3$ (in red) and $\tau = 21$ (in blue) for sample dates 2019-11-25 and 2019-11-07 for the expiry 2019-11-28 in the training period.

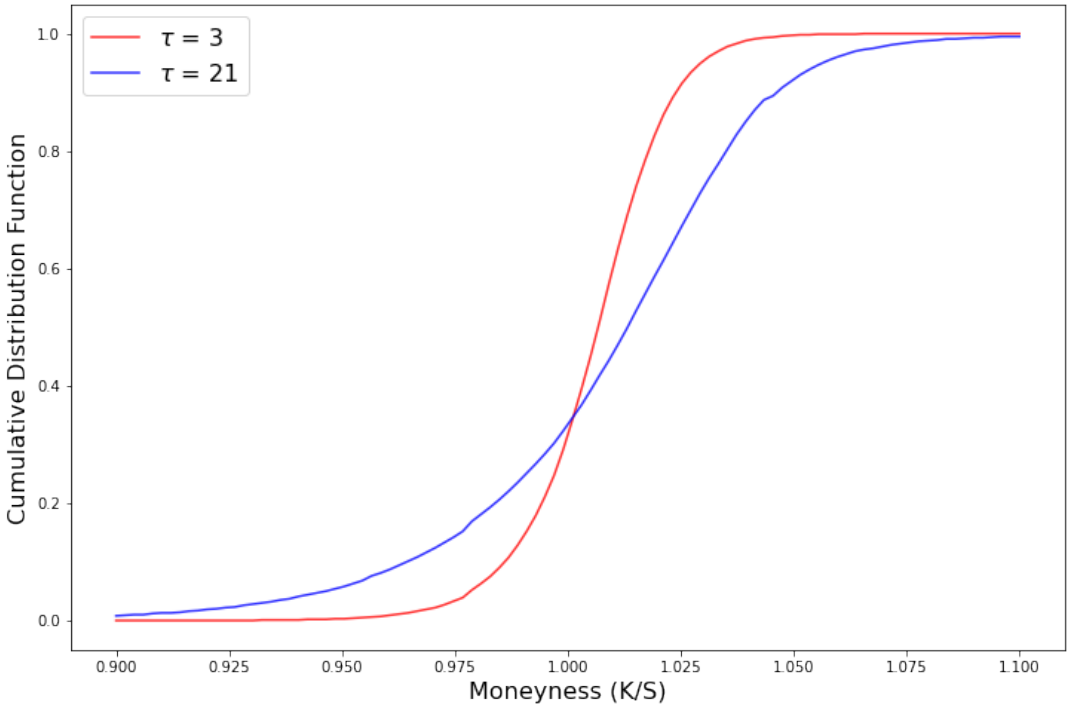


Figure 4: Change in risk-neutral cumulative distribution for $\tau = 3$

The figure shows the daily changes in the values of the risk-neutral cumulative distribution (in grey) along with the mean change (in red).

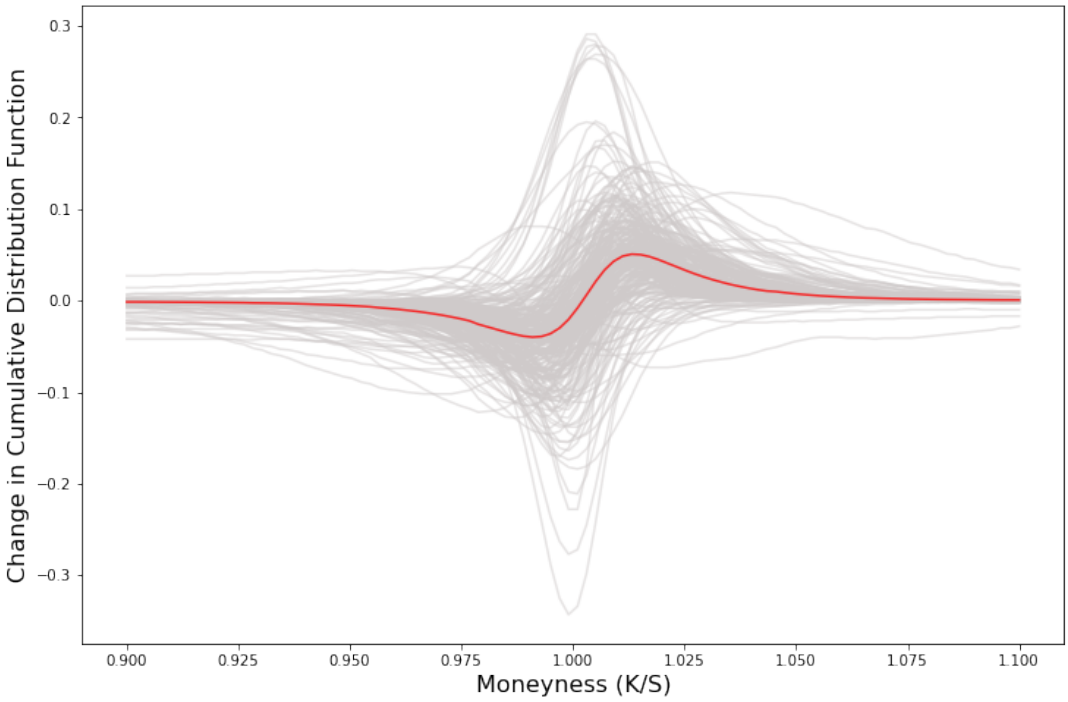


Figure 5: Simulated cumulative distributions for a sample date 2019-11-25.

The figure shows the simulated cumulative distributions (in grey) after applying the change values from 4 to the CDF in figure 3. The average distribution (is red) is also shown below.

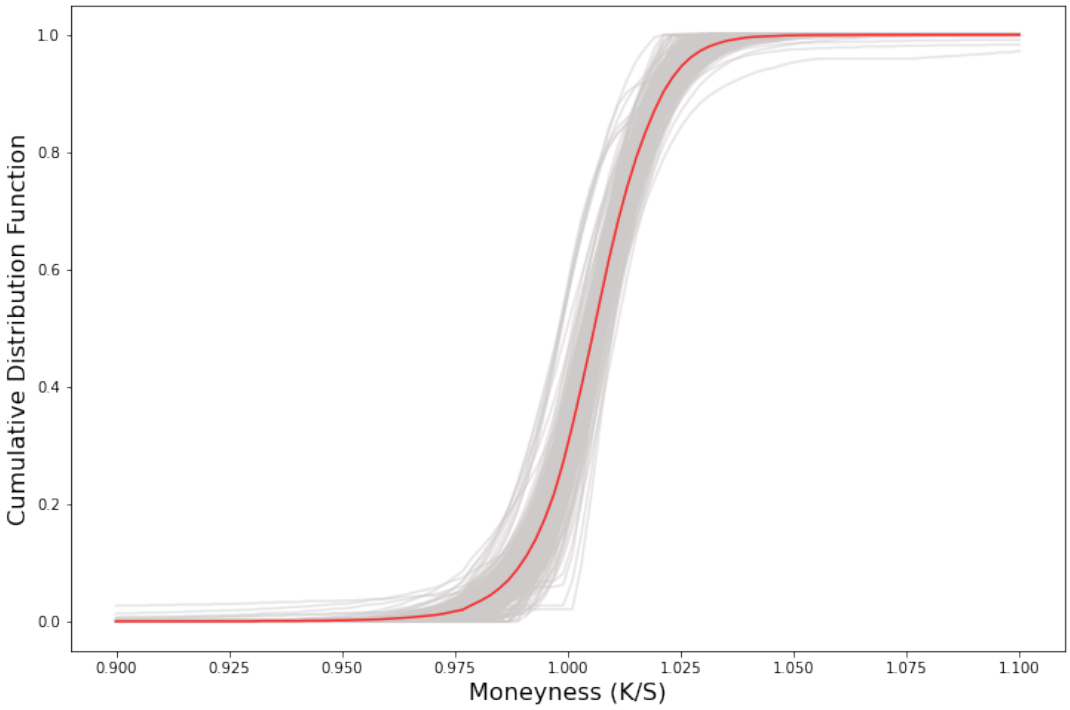


Figure 6: Comparison of distribution of returns

The figure shows the distribution of returns generated for a sample test date 2024-12-05 with the Monte-Carlo simulation (in blue) and TVAE (in red) returns along actual return value (in black).

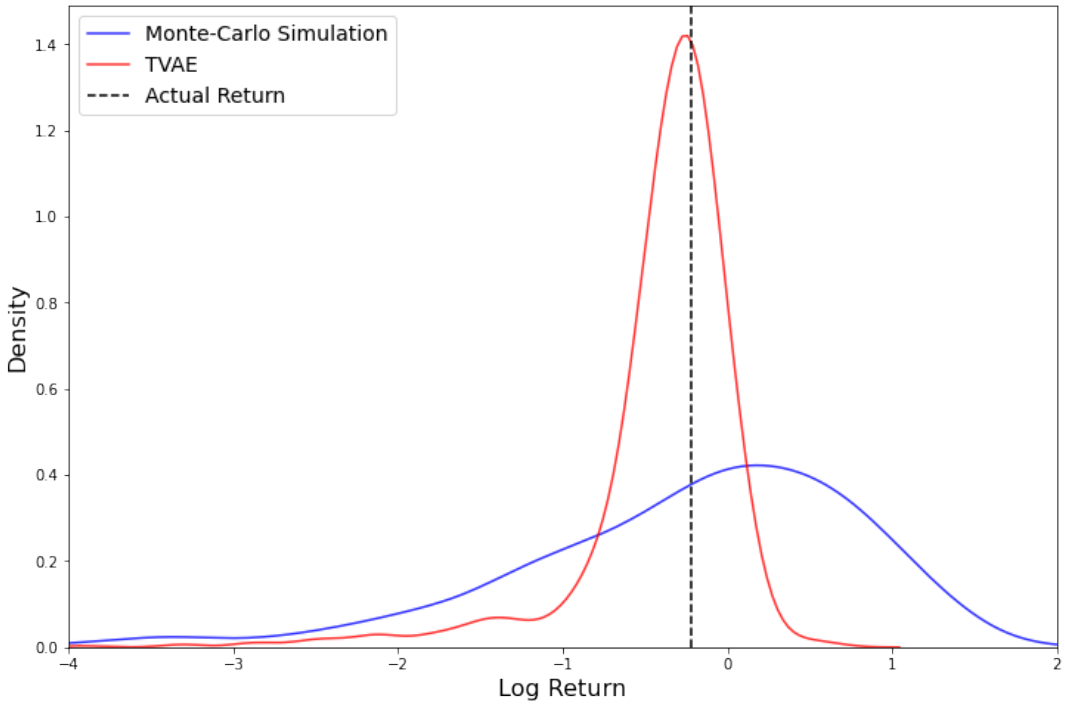
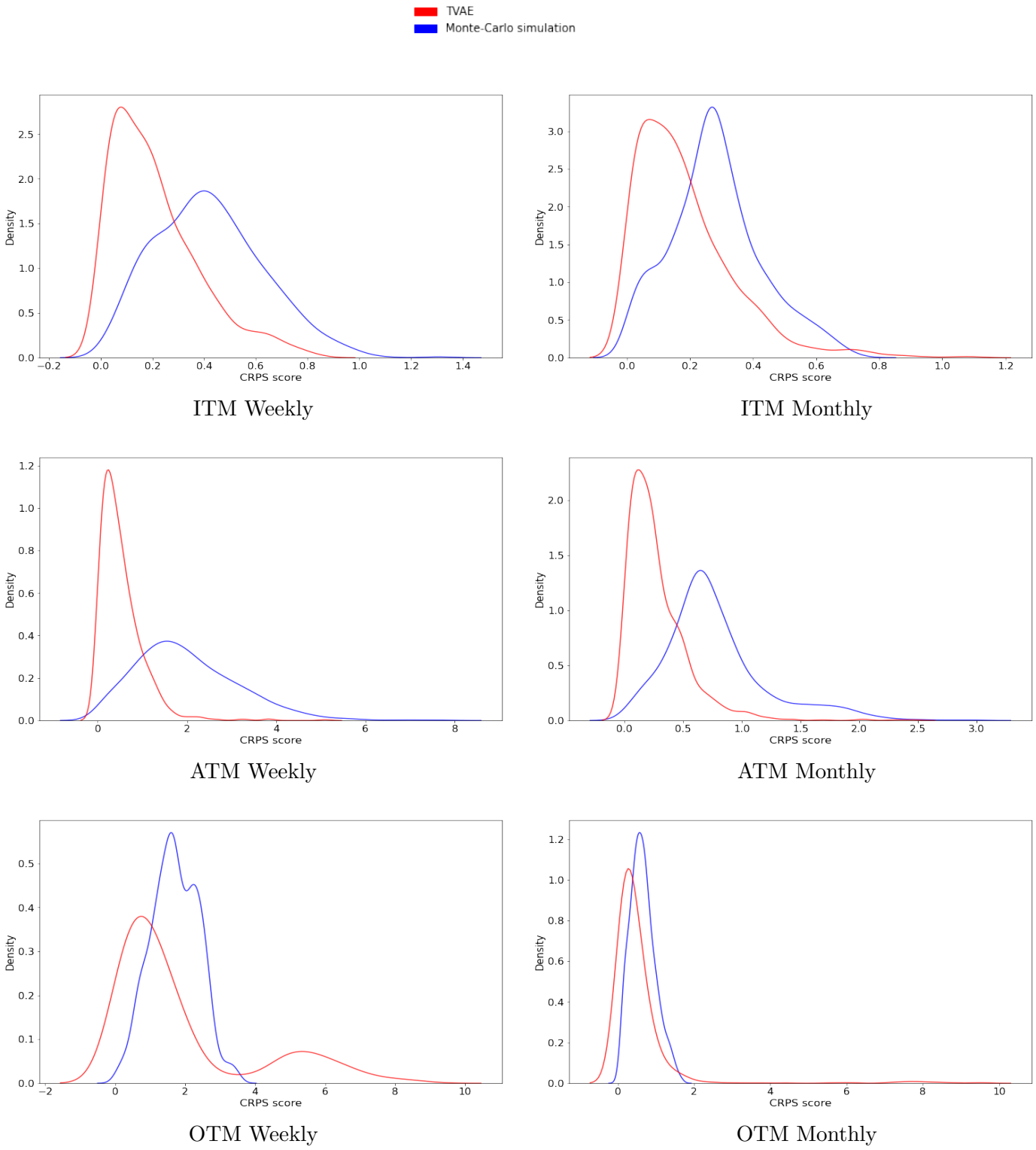


Table 1: CRPS Descriptive Statistics

The below table shows the descriptive statistics for the aggregated CRPS values for all the six buckets.

		Mean		Quantile 1		Median		Quantile 3	
Bucket	Maturity	MC	AE	MC	AE	MC	AE	MC	AE
ITM	Weekly	0.42	0.211	0.262	0.076	0.405	0.176	0.553	0.3
	Monthly	0.282	0.184	0.187	0.066	0.272	0.149	0.363	0.256
ATM	Weekly	1.995	0.525	1.178	0.199	1.808	0.404	2.682	0.717
	Monthly	0.776	0.275	0.528	0.099	0.682	0.212	0.921	0.379
OTM	Weekly	1.709	1.894	1.243	0.535	1.69	1.084	2.232	2.074
	Monthly	0.652	0.545	0.415	0.162	0.62	0.339	0.851	0.591
Overall		0.943	0.586	0.353	0.12	0.65	0.267	1.283	0.559

Figure 7: CRPS kernel-density estimate distributions
The below plots shows the distribution of CRPS values for all the six buckets.



ITM Weekly

ITM Monthly

ATM Weekly

ATM Monthly

OTM Weekly

OTM Monthly

Figure 8: Temporal nature of CRPS values

The below figure shows the temporal structure of CRPS values for all the six buckets.