Vanilla Deep Hedging

Study carried out by the Quantitative Practice
Special thanks to Mouad JALLOULI.

awalee

September 2023

SUMMARY

Introduction

The recent growth in hardware and data resources has made the application of machine learning techniques omnipresent in different fields. Machine learning has become an essential tool for statistical quants (asset allocation and alpha generation) on the buy side but it is shown to be also very prominent for hedging and derivative quants on the sell side.

One major application is the use of deep learning to hedge and price financial derivatives. Greek-based approach is the most common dynamic hedging strategy, it entails calculating the financial contract price sensitivities to variables and parameters of a given model. In the Black-Scholes (BS) framework, the sensitivities correspond to the partial derivatives of a partial differential equation (PDE) which is obtained in the continuous time limit $\Delta t \to 0$ of a delta-hedged vanilla contract. The elegance of BS PDE comes at the cost of non realistic assumptions such as the continuous re-hedging or the absence of transaction costs. However, in the "physical" option markets where the time step $\Delta t > 0$ is chosen according to the re-hedging frequency, the option pricing is argued to amount to an optimization problem that is similar to variational methods of Hamiltonian mechanics and stochastic optimal control in continuous time [1]. These alternative approaches, known in the literature as the global risk minimization, jointly optimize a sequence of hedging decisions with the objective of minimizing the expected value of a loss function applied to the terminal hedging error. With the emergence of deep learning, neural network emerges as a powerful tool to solve the above optimization problem. Recently, [2] applied this approach using a deep reinforcement learning (RL) algorithm in the presence of market frictions. The general framework of RL is for an agent to learn over many iterations of an environment how to select sequences of actions to optimize a cost function. The deep hedging algorithm trains an agent to learn how to approximate optimal hedging decisions by neural networks through many simulations of a market data. In addition, numerical results show that the training algorithm is able to effectively adapt hedging policies (i.e. neural networks parameters) to different stylized features of risky asset dynamics only by experiencing simulations of the financial market exhibiting these features.

1 Methodology

1.1 Theoretical Framework

The financial market is in discrete time of N timesteps with a finite time horizon of T years and known fixed trading dates:

$$\mathcal{T} := \{0 = t_0 < t_1 < \ldots < t_N = T\}$$

We consider $D+2$ liquid and tradable assets on the market with $D+1$ risky assets and one risk-free asset. Let $\{B_n\}_{n=0}^{N}$ be the price process of the risk-free asset with $B_n := \exp(rt_n)$ where $r \in \mathbb{R}$ is the annualized continuously compounded risk-free rate.

The risky assets include a non-dividend paying stock and D liquid vanilla European-type options such as calls and puts which expire on observation dates in $\mathcal{T}$.

Let $\{\bar{S}_{t_n}\}_{n=0}^{N}$ be the risky price process at the beginning of each trading period where $\bar{S}_{t_n} := \left[S^0_{t_n}, \ldots, S^D_{t_n}\right]$ with $S^0_{t_n}$ and $S^j_{t_n}$ being respectively the price of the underlying and of the j^{th} vanilla option.

1.2 Optimization Problem

The objective is to find a trading strategy $\delta := \{\delta_{t_n}\}_{n=0}^{N}$ to minimize our risk exposure to the derivative, where for $n = 1, \ldots, N$, $\delta_{t_n} := \left(\delta^{(0:D)}_{t_n}, \delta^{(B)}_{t_n}\right)$ is a vector containing the number of shares held in each asset of the hedging portfolio during the period $(t_{n-1}, t_n]$. $\delta^{(B)}_{t_n}$ and $\delta^{(0:D)}_{t_n} := \left[\delta^{(0)}_{t_n}, \ldots, \delta^{(D)}_{t_n}\right]$ are respectively the positions in the risk-free asset and in the $D+1$ risky assets. Furthermore, the initial portfolio (at time 0 before the first trade) is strictly invested in the risk-free asset. As each strategy δ_n is $\mathcal{F}_{t_n}$-adapted for all $0 \leqslant n \leqslant N-1$, the δ strategy is said to be adapted.

Moreover, let $\left\{V^{\delta}_{t_n}\right\}_{n=0}^{N}$ be hedging portfolio values for a trading strategy δ where $V^{\delta}_{t_n}$ is the value prior to re-balancing at time t_n :

$$V^{\delta}_{t_n} := \delta^{(0:D)}_{t_n} \cdot \bar{S}_{t_{n-1}} + \delta^{(B)}_{t_n} \cdot B_{t_n}, \quad n = 1, \ldots, N,$$

and $V^{\delta}_0 := \delta^{(B)}_0$ since the initial capital amount is assumed to be strictly invested in the risk-free asset.

At maturity T, the value of the hedging portfolio is:

$$V^{\delta}_T = V^{\delta}_0 + H^{\delta}_N$$

where:

- V^{δ}_0 is the price the hedger charge at inception to risk manage the contract till maturity T using the strategy δ.
- $H^{\delta}_N := \sum_{k=1}^{N} \delta^{(0:D)}_{t_k} \cdot \left(\bar{S}_{t_k} - \bar{S}_{t_{k-1}}\right)$ is the wealth accumulated from the strategy δ through buying and selling the hedging instruments to neutralize risk factors.

Note that the optimization problem solve jointly for the price and the hedging strategy (δ, V^{δ}_0). A simpler approach would be to take the price as an exogenous input and solve for the optimal strategy δ only. We adopt the simple approach and solve directly for $\delta^{*}(V_0)$ that minimizes the hedging error

$$Z_T - V^{\delta}_T = Z_T - \left(V_0 + H^{\delta}_N\right)$$

V_0 is given as an input and doesn't depend on δ anymore.

Z_T is the financial contract payoff at maturity.

1.3 Approximating the optimal strategy by a neural network

The role of neural network is to approximate the complex functional that maps a contract to an optimal replication strategy. Depending on the use case, several neural network architectures can be used but they are all derived from the basic feed forward neural network (FFNN).

1.3.1 Architecture

Feed Forward Neural Network

Let $d_0, d_1, \ldots, d_L, d \in \mathbb{N}$, a feed forward neural network $F_\theta : \mathbb{R}^{d_0} \to \mathbb{R}^d$,with L hidden layers of dimension d_l and input and output layers of dimensions d_0, d respectively, is defined as:

$$F_\theta(X) := o \circ f_L \circ \ldots \circ f_1$$

$$f_l(X) := g(W_l X + b_l), \quad l = 1 \ldots, L,$$

where $W_l \in \mathbb{R}^{d_l \times d_{l-1}}$ and $b_l \in \mathbb{R}^{d_l \times 1}$ represent the weight and the bias vector of the l^{th} layer and $g : \mathbb{R} \to \mathbb{R}$ is a non-linear function that represents the activation function.

$o : \mathbb{R}^{d_L} \to \mathbb{R}^d$ is the output function which applies an affine transformation to the output of the last hidden layer h_L and possibly also a nonlinear transformation with the same range as F_θ depending on the output nature. The trainable parameters θ is the set of all weight matrices and bias vectors which are learned by minimizing a specific cost function. Hence, the optimal strategy is equivalent to an optimal set of parameters $\theta^\star$:

$$\delta^\star = \delta^{\theta^\star}$$

Recurrent Neural Network

Recurrent Neural Networks (RNNs) are derived from a FFNN by introducing a feedback loop: each hidden layer is a function of both an input vector from the current time-step and an output vector from the hidden layer of the previous time-step, which enables to exploit the temporal order of the input data. More formally, as shown in figure 2, for an input vector X_{t_n} at time t_n, the time-t_n output of the hidden layer is computed as $h_{t_n} = f(h_{t_{n-1}}, X_{t_n})$ for some time-independent function $f(W, U)$.

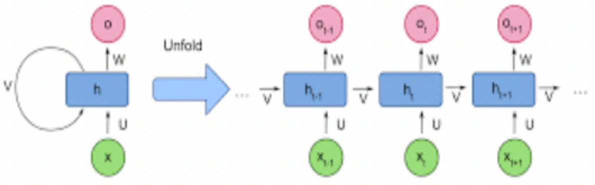


Figure 2: RNN architecture

The feature vector at each time-step (re-balancing date) t_n is :

$$X_{t_n} := \left[\log\left(\bar{S}_{t_n}\right), T - t_n, \frac{V^\delta_{t_n}}{V^\delta_0} \right], \quad n = 0, \ldots, N-1,$$

and F_θ outputs at each time step the position to hold in each of the $D+1$ instruments:

$$O_{t_{n-1}} = \left[\delta^{(0)}_{t_n}, \ldots, \delta^{(D)}_{t_n}\right]$$

The input X_{t_n} is a $D+3$ dimensional vector which contains, in the same order, the logarithm of the $D+1$ assets prices, the time-to-maturity (TTM) and the normalized hedging portfolio value.

Note that RNNs are usually used in supervised learning paradigm, but in the context of deep hedging the RNN is never explicitly told, during its training phase, the right hedging decision at a given time-step. Instead, the RNN must learn through trial-and-error from several simulations to dynamically adapt its embedded policy (i.e trainable parameters) with the objective of minimizing the expected hedging error at the final time-step. It is actually a reinforcement learning approach using RNN and not a supervised learning approach.

1.3.2 Loss function

The role of the neural network is to approximate the optimal strategy function by finding the optimal set of parameters θ that minimize the hedging error $Z_T - V^\delta_T$. The hedging error is a stochastic variable given the the stochastic nature of the training dataset. But to solve the minimization problem, a deterministic quantity is needed. This quantity must be coherent with the human risk perception (e.g the risk of two portfolios together cannot get any worse than adding the two risks separately). Measures satisfying this criteria are called coherent risk measures. Usually a more general class of measures called convex risk measures are used.

We choose to quantify our stochastic hedging error in an $\mathbf{L}^2(\mathbb{P})$ space using the quadratic loss function which penalizes equally losses and gains.

The optimization problem is then formulated as:

$$\delta^\star(V_0) := \arg\min_\delta \mathbb{E}\left[\left(Z_T - (V_0 + H^\delta_N)\right)^2\right]$$

where V_0 is the given contract premium.

The premium V_0 is supposed to be the contract's fair hedging price. In case it is unknown, the optimal premium $V_0^\star$ will be the one minimizing [3] the function:

$$V_0 \to \mathbb{E}\left[\left(Z_T - (V_0 + H_N^{\delta(V_0)})\right)^2\right]$$

Then the pair $\left(V_0^\star, \delta(V_0^\star)\right)$ solves the joint optimization problem of finding an optimal strategy δ and its premium V_0^δ.

Moreover, the derivative of the function is null at the optimal point $V_0^\star$:

$$\mathbb{E}\left[Z_T - V_0^\star - H_N^{\delta(V_0^\star)}\right] = 0$$

Hence, the final profit and loss (PnL) distribution is expected to be centered around 0 if the given V_0 is the right fair price. Otherwise, the distribution would be centered around the difference between the given input price and the true fair price. The fair price is then the byproduct of solving for the optimal hedging strategy regardless of the initial capital.

Note that, the adopted approach is a basic mean-variance optimization. But many other convex risk measures can be considered depending on the risk aversion of the hedger. Conditional Value-at-Risk (CVaR) is such an example related to the more general quantile hedging approach [4].

2 Application to Black-Scholes framework

2.1 Assumptions

The underlying stock S_{t_n} is the only stochastic required to generate the inputs X_{t_n}. In the BS framework, the underlying is assumed to follow a geometric brownian motion (GBM). A sample of 100.000 simulated discrete GBM paths are generated as:

$$S_t = S_0 \exp\Big(-\frac{1}{2}\sigma^2 t + \sigma W_t\Big)$$

with S_0 is the initial stock price, and $N = 60$ timesteps. The real market drift is assumed null. The interest and dividend rates are both taken equal to zero. The pricing is under the risk neutral probability. For numerical results, we consider the simple case of hedging a short vanilla Put option, in the BS world (GBM dynamics with a constant volatility), by taking a short position in the underlying. The payoff reads $Z_T = max(K - S_T, 0)$ with K the strike value. In this case $D = 0$ as the underlying is the only required hedging instrument. This strategy corresponds to delta-hedging a financial contract which is the argument behind the Black-Scholes risk-neutral valuation. The trader takes position (sell or buy) in the underlying at a given frequency each time step (e.g $\Delta t = 0.5day, 1day\dots$).

The RNN results are bench-marked against the analytical BS delta-hedging formulas. The analytical BS delta is calculated as:

$$\Delta = N\left[\frac{log(\frac{F}{K}) + \frac{1}{2}\sigma^2(T-t)}{\sigma\sqrt{T-t}}\right] = \frac{\partial Z}{\partial S}$$

with $x \to N(x)$ the cumulative distribution function (cdf) of the standard normal distribution, F the forward value, and $T - t$ the time to maturity (TTM). In practice, the volatility term σ corresponds to the BS implied volatility (IV) which is meant to be equal to the underlying realized volatility (RV) through the GBM dynamics.

The difference between the two volatilities affects the final PnL of a delta-hedged vanilla option portfolio value. The BS robustness formula quantifies this gamma PnL leak as:

$$PnL_T = \int_0^T \frac{1}{2} S_t^2 \big(IV - RV\big) \Gamma_t^{IV}(t, S_t)\, dt$$

The gamma $\Gamma_t^{IV}(t, S_t)$ measures the the second order sensitivity of the contract to the underlying:

$$\Gamma^{IV} = \frac{\partial \Delta^{IV}}{\partial S} = \frac{\partial^2 Z}{\partial S^2}$$

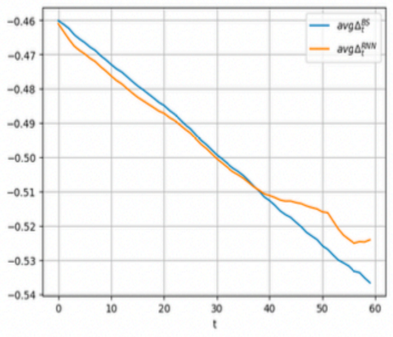
The error is due to using the $\Delta^{IV} = \Delta(\sigma = IV)$ for delta-hedging instead of the true volatility delta $\Delta^{RV} = \Delta(\sigma = RV)$. Note that this error cannot be eliminated even when delta-hedging continuously as assumed by BS framework.

2.2 Numerical Results

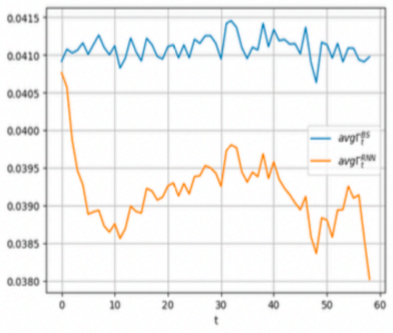
The case IV = RV

Here, we assume the realized volatility used to generate the synthetic market data (GBM) is equal to the implied volatility used to compute the BS delta ($IV = RV = 20\%$). In this case, the hedging error is due only to the discrete re-balancing.

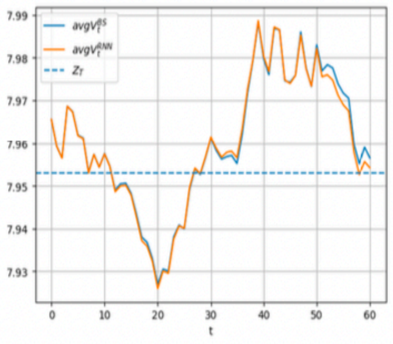
On average, the two strategies are similar as shown in figure 3, the RNN hedging portfolio (figure 5.c) is on average closer to the final payoff at maturity Z_T. This means that the BS gamma PnL is higher than the RNN gamma PnL which is due to the gamma of the BS delta being higher than the RNN gamma (figure 5.b).



(a) Delta over time



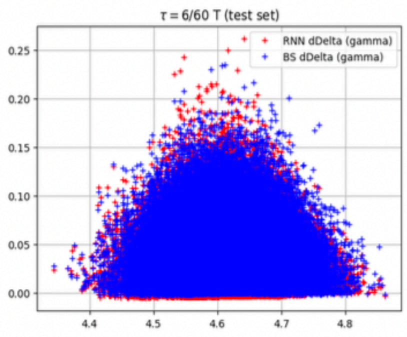
(b) Gamma over time



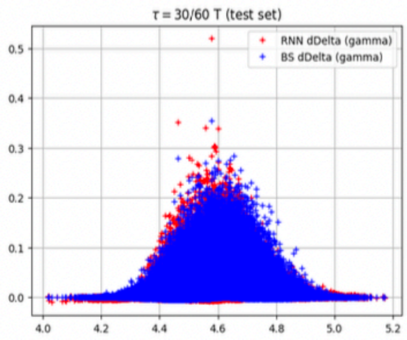
(c) Hedging portfolio value over time vs average payoff Z_T (premium $P = 7.95\%$).

Figure 3: Average RNN strategy over time

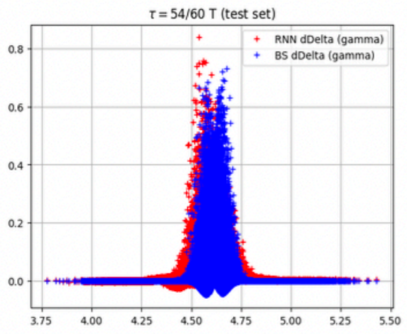
A lower gamma means that the delta change (between consecutive re-balancing dates) is lower with a RNN strategy. This observation is shown in figure 4 which includes the gamma bleed (the quantity $d\Delta = \Gamma dS$ is plot instead of Γ).



(a) TTM = 54 days



(b) TTM = 30 days



(c) TTM = 6 days

Figure 4: Gamma bleed

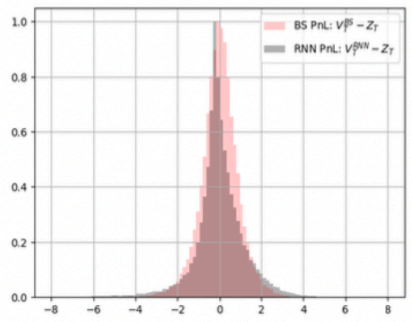
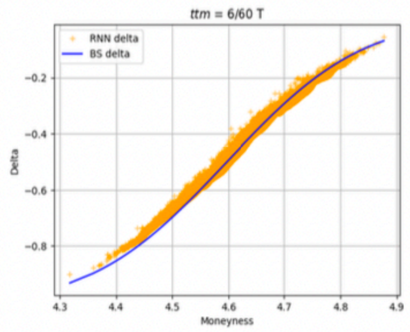
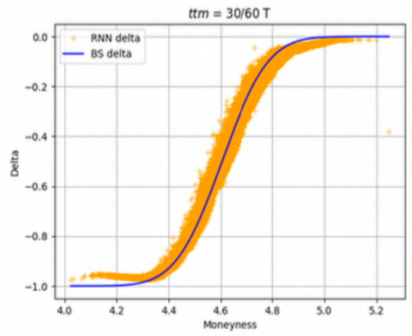


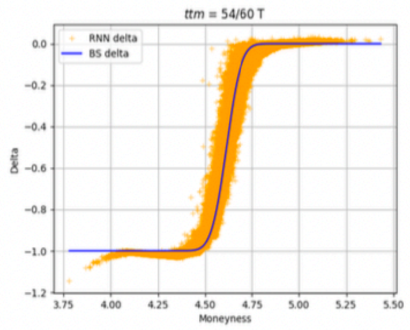
Figure 5: PnL histogram IV=RV



(a) TTM = 54 days



(b) TTM = 30 days



(c) TTM = 6 days

Figure 6: Delta bleed $IV = RV$

The actual delta is shown in figure 8 (more in figure 9). The RNN delta matches the BS delta overall with a slight difference far from the money which explains the observed gamma PnL discrepancy. Note that, the RNN cdf (in orange) corresponds more to a fat-tailed distribution's cdf such as the t-distribution than a BS gaussian cdf (in blue). This explains the final RNN PnL distribution being fatter compared to the BS PnL distribution as shown in figure 5. One explanation is that the discontinuous delta-hedging might introduce jumps between two re-balancing dates. Jump-diffusion models in general lead to fat-tailed PnL distribution. Using risk measures that penalize extreme losses such as CVaR is required in order to tackle this issue.

To conclude, the RNN delta strategy manages to converge to the BS analytical delta strategy while adjusting for the discontinuous re-balancing constraint. While this constraint is very realistic, it is not taken into account in the BS assumptions which makes the RNN strategy more practical.

The case IV ≠ RV

In this setting, we consider an even more realistic scenario where the implied volatility ($IV = 40\%$) is different from the realized volatility ($RV = 20\%$).

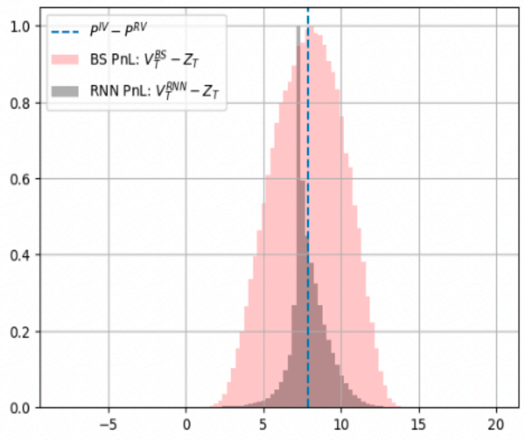


Figure 7: PnL histogram IV ≠ RV

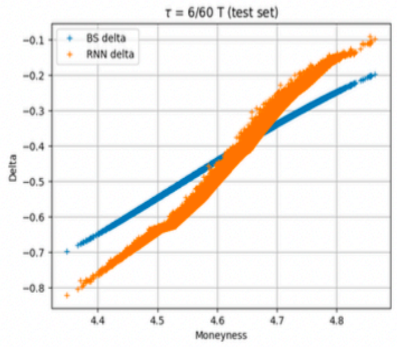
Figure 7 shows the distribution of the final PnL. The hedging error is centered around the difference between the IV BS price P^{IV} (charged price) and RV BS price P^{RV} (true fair price). This positive PnL profit is due to overpricing the contract. The actual variance-optimal hedging fair price is:

$$V_0^\star = |\mathbb{E}[PnL_T] - V_0| = P^{IV} - (P^{IV} - P^{RV}) = P^{RV}$$

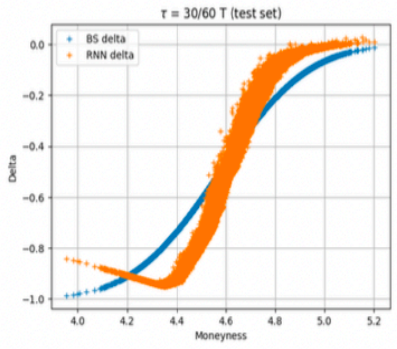
Re-training the RNN with the deduced fair price $V_0^\star$ will shift the PnL distribution (1st moment) back to zero while keeping the same shape (higher moments).

Moreover, the RNN gamma PnL is actually similar to the gamma PnL seen in the case IV = RV (figure 5) which is due only to the discontinuous re-balancing. Indeed, the RNN naturally hedge with the RV because its IV is simply the estimated volatility from the realized simulation paths.

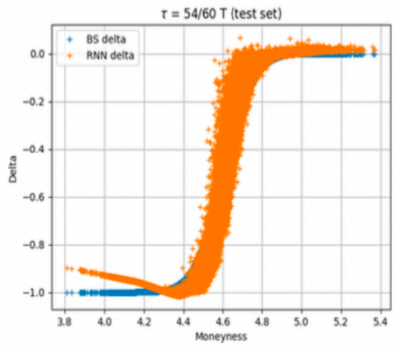
On the other hand, the BS gamma PnL is clearly fatter, as it combines the discontinuous re-balancing with the constraint $IV > RV$. Indeed, the difference $IV - RV$ amplifies the discontinuous re-balancing hedging error as explained, earlier, by the BS robustness formula. Unless the $IV \approx RV$, this PnL cannot be reduced by re-hedging more frequently. The RNN delta hedging has a clear practical advantage in this case. However, increasing the frequency of hedging will increase the size and the number of RNN timesteps which will make the training more challenging as the architecture gets bigger.



(a) TTM = 54 days



(b) TTM = 30 days



(c) TTM = 6 days

Figure 8: Delta bleed with $IV > RV$

Conclusion

The deep hedging algorithm is capable of retrieving the analytical BS delta hedging while adjusting for non-practical assumptions such as the continuous hedging or the implied volatility being equal to the future realized volatility.

The deep hedging PnL is independent of any assumption about the RV through an IV. The frequency of re-hedging is however a constraint for the deep hedging approach.

Although the deep hedging approach is model-independent (free greek approach), it still relies on the the training dataset. Training on real market data which is representative of all possible regime switches might make the approach totally model-independent. However, the limitation is the absence of enough historical market data to use as a training data source. Hence, the need for generative deep learning methods to synthesize enough real training data.

An additional constraint would be the regulation requirements regarding the explainability of the appraoch.

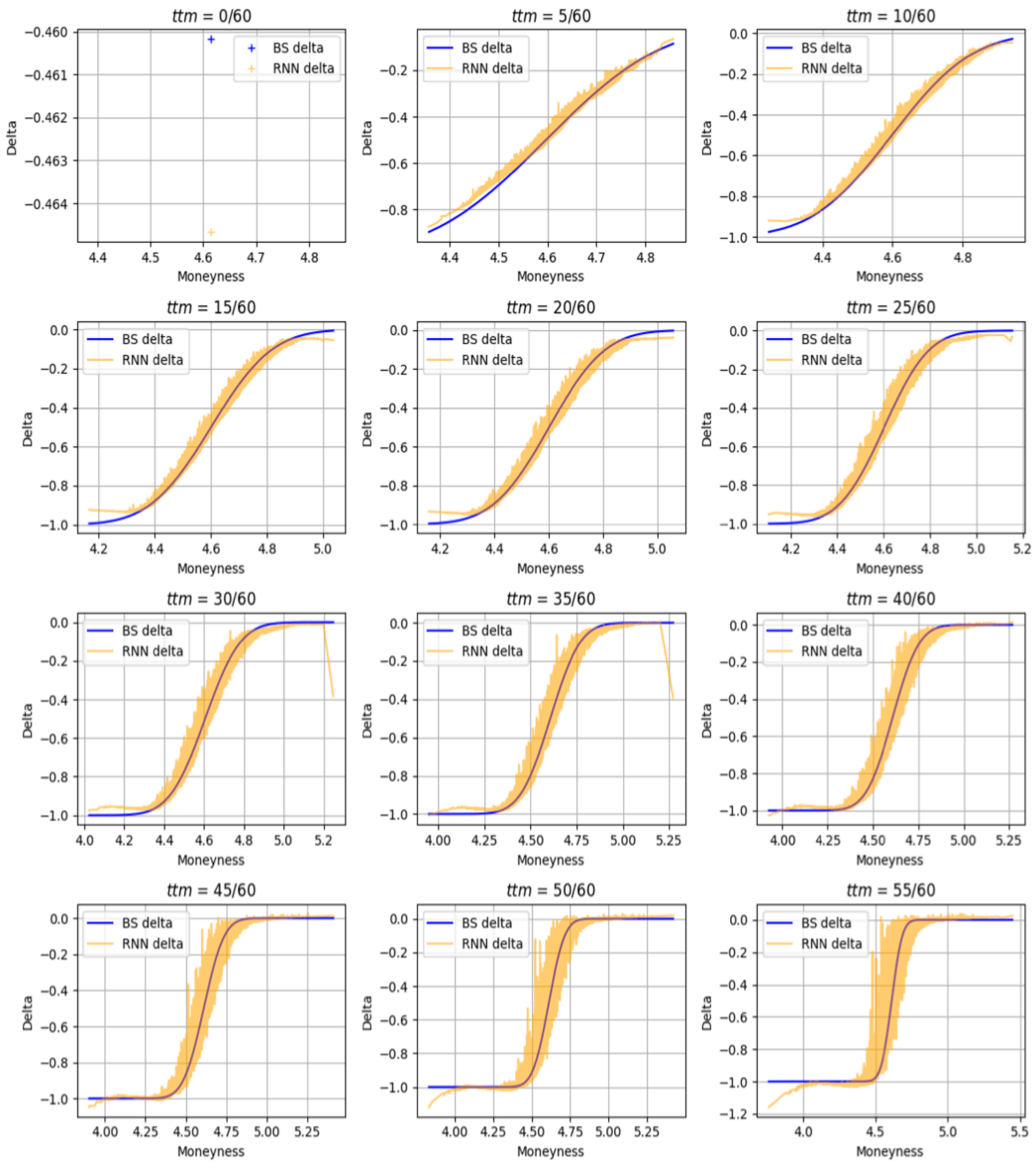


Figure 9: Delta bleed comparison IV=RV

References

[1] Halperin Igor. Qlbs: Q-learner in the black-scholes (-merton) worlds. *SSRN: https://ssrn.com/abstract=3087076*, 2017.

[2] Hans Bühler, Lukas Gonon, Josef Teichmann, and Ben Wood. Deep hedging, 2018.

[3] M Schweizer. Variance-optimal hedging in discrete time. *Mathematics of Operations Research*, 20(1):1–32, 1995.

[4] Peter Föllmer, Hans & Leukert. Quantile hedging. *Humboldt University of Berlin, Interdisciplinary Research Project*, 373(13):1–24, 1998.

A propos d'Awalee

Cabinet de conseil indépendant spécialiste du secteur de la Finance.

Nous sommes nés en 2009 en pleine crise financière. Cette période complexe nous a conduits à une conclusion simple : face aux exigences accrues et à la nécessité de faire preuve de souplesse, nous nous devions d'aider nos clients à se concentrer sur l'essentiel, à savoir leur performance.

Pour accomplir cette mission, nous nous appuyons sur trois ingrédients : habileté technique, savoir-faire fonctionnel et innovation.

Ceci au service d'une ambition : dompter la complexité pour simplifier la vie de nos clients.

«*Run the bank*» avec Awalee !

Contactez-nous

Ronald LOMAS
Partner
rlomas@awaleeconsulting.com
06 62 49 05 97

est une marque de