# Computer Assignment 2

Stochastic Processes: the Fundamentals
Vrije Universiteit Amsterdam

| | |
|---|---|
| Name: | Andreas Koulopoulos, Milan Peter |
| Student Nr.: | 2897428, 2868506 |
| Faculty: | School of Economics and Business |
| Supervisor: | Prof. Mark-Jan Boes |

## 1 Monte Carlo Simulation

### 1.a Calculate the price of a European call option with Binomial Trees and BS

The analysis begins with parameter estimation using historical S&P 500 data, processed and computed in Computer Assignment 1. The volatility of the sample is computed as 0.045373 on a monthly basis, which, when annualized using the standard square root scaling rule, yields the following.

$$\sigma = \sqrt{12} \times 0.045373 = 0.157175.$$

The risk-free rate is specified as 3% with quarterly compounding, which requires conversion to continuous compounding using the logarithmic transformation.

$$r_{\log} = \ln(1.03) = 0.0296,$$

to maintain consistency with the continuous-time Black–Scholes framework. The current stock price is set at $S_0 = \$6460.26$, the strike price at $K = \$6500$, and the time to maturity at $T = 0.25$ years, representing a 3-month European call option.

The binomial tree method represents the first numerical approach examined, implemented with 300 time steps to provide a discrete approximation to the continuous geometric Brownian motion process. The theoretical foundation of this method is based on the risk-neutral valuation principle, where the option price is equal to the expected discounted payoff under the risk-neutral measure.

The up and down factors are calculated as

$$u = e^{\sigma\sqrt{\Delta t}} \quad \text{and} \quad d = \frac{1}{u},$$

where $\Delta t = \frac{T}{n}$ represents the size of the time step.

The Risk-Neutral Probability

$$q = \frac{e^{r\Delta t} - d}{u - d}$$

ensures that the expected return on the stock equals the risk-free rate under the risk-neutral measure. The implementation achieves a call option price of \$206.57, which represents an error of only \$0.12 compared to the analytical solution Black–Scholes, or a relative error of approximately 0.06%.

## 1.b Monte Carlo Simulation with Euler Discretization

We implemented the Monte Carlo simulation using the Euler discretization scheme to estimate European call option prices under geometric Brownian motion (GBM).

**Euler Discretization.** For the stochastic differential equation (SDE)

$$dS_t = rS_t\,dt + \sigma S_t\,dW_t,$$

the Euler discretization scheme is given by

$$S_{t+\Delta t} = S_t + rS_t\Delta t + \sigma S_t\sqrt{\Delta t}\,Z,$$

where $Z \sim \mathcal{N}(0,1)$ are independent standard normal random variables.

**Implementation Details.**

- Time horizon: $T = 0.25$ years (3 months)
- Time steps: $N = 3$ (monthly steps, $\Delta t = 1/12$ year)
- Sample sizes: $M \in \{100, 500, 1000, 5000, 10000\}$
- Parameters: $r = 0.0296$, $\sigma = 0.1572$, $S_0 = 6460.26$, $K = 6500$

**Monte Carlo Procedure.**

1. For each simulation path $m$:
   - Initialize $S_0 = 6460.26$.
   - Apply the Euler scheme for three monthly steps to obtain $S_T$.
2. Compute the option payoff as $\max(S_T - K, 0)$.
3. Discount the expected payoff at the risk-free rate to obtain the option price.

The results demonstrate the impact of sampling error on Monte Carlo estimates, with prices varying from \$219.81 for 100 simulations to \$209.07 for 10,000 simulations. The convergence behavior follows the expected $1/\sqrt{M}$ rate, where $M$ represents the number of simulation paths, indicating that quadrupling the number of simulations approximately halves the standard error. However, the persistent bias of approximately \$2.62 above the Black–Scholes price reveals the presence of discretization error inherent in the coarse time stepping.

## 1.c Monte Carlo Simulation with Euler Discretization (Dt = 1 day)

The Monte Carlo simulation using the Euler discretization method was implemented to examine the impact of time step size on option valuation accuracy. The reference price for Black–Scholes is \$206.45, while the Euler method produced an option value of \$209.07 with a coarse time step ($\Delta t = 1/12$ year) and \$203.79 with a finer time step ($\Delta t = 1/63$ year). The corresponding deviations from the analytical benchmark are +\$2.62 (+1.27%) and –\$2.65 (–1.28%), respectively.

Reducing the time step from one month to one day significantly improves accuracy, demonstrating the expected reduction in discretization bias. This improvement aligns with the first-order convergence property of the Euler scheme, where the error is proportional to the step size $\mathcal{O}(\Delta t)$. Smaller time increments provide a more accurate approximation of the continuous geometric Brownian motion by capturing finer fluctuations in the simulated price paths.

The coarse discretization overestimates the option price due to missing intra-period variance, while the fine discretization slightly underestimates it, likely reflecting convexity effects in the exponential transformation of simulated returns. The daily-step implementation, assuming 63 trading days per quarter, thus yields results much closer to the continuous-time Black–Scholes benchmark.

## 1.d Monte Carlo Simulation with Euler Discretization (Dt = 1 day) on LogS

A Monte Carlo simulation was implemented using the Euler discretization of the logarithmic price process, which provides a more accurate representation of geometric Brownian motion than direct price simulation. The log-price evolves according to

$$d(\ln S) = \left(r - \tfrac{1}{2}\sigma^2\right) dt + \sigma\, dW,$$

where $\Delta t$ corresponds to one trading day. Simulating $\ln S$ eliminates the discretization bias inherent in direct price paths and ensures that all simulated prices remain positive. The resulting option prices for different numbers of simulation paths ($M$) show clear convergence toward the Black–Scholes benchmark. For $M = 100$, 500, 1000, 5000, and 10000, the estimated call prices are \$209.07 with differences from the Black–Scholes value of \$-25.85, \$9.37, \$8.64, \$-0.09, and \$-2.62, respectively. The results indicate that accuracy improves substantially as $M$ increases, with the log-price simulation producing results closely aligned with the analytical Black–Scholes value while minimizing numerical instability.

## 1.e Crank–Nicolson finite differences

The third numerical method applied is the Crank–Nicolson finite difference scheme, which directly solves the Black–Scholes partial differential equation,

$$\frac{\partial V}{\partial t} + rS\frac{\partial V}{\partial S} + \tfrac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} - rV = 0,$$

by discretizing both time and asset price dimensions. The spatial grid ranges from $S_{\min} = 0$ to $S_{\max} = 3S_0$, divided into $M$ price steps and $N$ time steps, with $\Delta S = (S_{\max} - S_{\min})/M$ and $\Delta t = T/N$. The Crank–Nicolson scheme averages the explicit and implicit finite-difference formulations, resulting in a tridiagonal system at each time step:

$$AV^{n+1} = BV^n + \text{boundary terms},$$

where $A$ and $B$ are tridiagonal matrices containing coefficients $\alpha_j = \frac{1}{4}\Delta t(\sigma^2 j^2 - rj)$, $\beta_j = -\frac{1}{2}\Delta t(\sigma^2 j^2 + r)$, and $\gamma_j = \frac{1}{4}\Delta t(\sigma^2 j^2 + rj)$. The system is solved iteratively backward from maturity, enforcing the terminal condition $V(S,T) = \max(S-K, 0)$ and boundary conditions $V(0,t) = 0$ and $V(S_{\max}, t) = S_{\max} - Ke^{-r(T-t)}$.

For $(M, N) = (400, 400)$, the computed call price is \$206.54, differing only by \$0.09 (0. 05%) from the analytical Black–Scholes value of \$206.45, which confirms the numerical convergence of order $\mathcal{O}(\Delta S^2, \Delta t^2)$. Compared to the Monte Carlo and binomial methods, the Crank–Nicolson approach achieves higher accuracy with deterministic convergence and no stochastic noise, making it the preferred method for approximating the true option price in this setting.

### 1.f Gap Call Option pricing with Euler Discretization

The value of the gap call option was estimated using the Euler discretization of the geometric Brownian motion under the risk-neutral measure. The logarithm of the stock price was simulated according to

$$\log S_{t+\Delta t} = \log S_t + \left(r - \frac{1}{2}\sigma^2\right)\Delta t + \sigma\sqrt{\Delta t}\, Z_t,$$

where $Z_t \sim \mathcal{N}(0,1)$, with a time step of one trading day ($\Delta t = T/63$) and $M = 10{,}000$ simulated paths. At maturity, each terminal price $S(T)$ was used to compute the gap call payoff

$$\text{Payoff} = \begin{cases} S(T) - K_1, & \text{if } S(T) > K_2, \\ 0, & \text{otherwise,} \end{cases}$$

where $K_1 = 6600$ and $K_2 = 6500$. The Monte Carlo estimator for the option value is

$$C_{\text{gap}} = e^{-rT}\,\mathbb{E}[\max(S(T) - K_1, 0) \cdot \mathbf{1}_{\{S(T)>K_2\}}].$$

Using $S_0 = 6460.26$, $r = 0.0296$, $\sigma = 0.1572$, and $T = 0.25$, the simulation produced a convergent price of approximately \$155.84 for $M = 10{,}000$. This value is lower than the regular call price of \$206.57 obtained from the Black–Scholes model because the gap call yields zero when $S(T)$ lies between $K_2$ and $K_1$. Hence, the gap call sacrifices partial upside potential in exchange for maintaining the same activation threshold, making it strictly less valuable than the standard European call.

## 2 Dynamic Hedging

### 2.a Option Valuation under the Black-Scholes Model

In the second section, we are valuing a European call option on the S&P 500 index under the **Black-Scholes model**. Given parameters:

$$\begin{aligned} S_0 &= 6{,}500 && \text{(current index level)} \\ K &= 6{,}500 && \text{(strike price)} \\ r &= 0.029559 && \text{(continuously compounded risk-free rate)} \\ \sigma &= 0.0453 && \text{(volatility estimate from Part I)} \\ T &= 0.25 && \text{(time to maturity in years, i.e., 3 months)} \end{aligned}$$

The **Black-Scholes formula** for a European call option is given by:

$$C = S_0 N(d_1) - K e^{-rT} N(d_2),$$

where

$$d_1 = \frac{\ln(S_0/K) + (r + 0.5\sigma^2)T}{\sigma\sqrt{T}}, \qquad d_2 = d_1 - \sigma\sqrt{T}.$$

Substituting the given values:

$$d_1 = 0.1335, \qquad d_2 = 0.0551.$$

Hence,

$$C = 6{,}500 \times N(0.1335) - 6{,}500 \times e^{-0.029559\times 0.25} \times N(0.0551),$$

which yields

$$C = 227.2553 \text{ USD per index unit.}$$

Since each option contract controls 100 index units and the trader sells 20 contracts, the total cash inflow is:

$$\text{Total proceeds} = 227.2553 \times 100 \times 20 = 454{,}510.60 \text{ USD.}$$

The trader receives USD 454,510.60 upfront for selling the 20 call option contracts. This amount represents the **fair value** of the options in a **perfect Black-Scholes world**, where no arbitrage opportunities exist and markets are frictionless.

The price incorporates:

- the *intrinsic value* (which is negligible as $S_0 = K$), and
- the *time value* arising from volatility ($\sigma = 4.53\%$) and the 3-month time to maturity.

The proceeds form the initial cash inflow of the trader's **short call position**, which must later be **dynamically hedged** to maintain a risk-neutral portfolio in subsequent parts of this exercise.

## 2.b Dynamic Delta Hedging (Weekly) in Black–Scholes

We replicate a long European call by dynamically trading the index and cash, rebalancing *weekly* over 13 weeks, and evaluating the trader's P&L from shorting the call and running the hedge.

We assume a perfect Black–Scholes world with geometric Brownian motion (GBM):

$$\frac{dS_t}{S_t} = r\,dt + \sigma\,dW_t, \qquad S_0 = 6{,}500, \;\; K = 6{,}500, \;\; T = \tfrac{1}{4} \text{ year}, \;\; r = 0.029559, \;\; \sigma = 0.0453.$$

Discretization (weekly, $\Delta t = T/13$):

$$S_{t+\Delta t} = S_t \exp\Big((r - \tfrac{1}{2}\sigma^2)\Delta t + \sigma\sqrt{\Delta t}\,Z\Big), \qquad Z \sim \mathcal{N}(0,1).$$

At rehedge date $t_i$ (weekly grid), compute Black–Scholes delta of the *long* call:

$$\Delta_i = N(d_{1,i}), \qquad d_{1,i} = \frac{\ln(S_{t_i}/K) + \big(r + \frac{1}{2}\sigma^2\big)(T - t_i)}{\sigma\sqrt{T - t_i}}.$$

To replicate a *long* call, hold $\Delta_i$ units of the index and $(B_i)$ in the cash account at rate $r$ so that the hedge portfolio value matches the call value $C(S_{t_i}, t_i)$:

$$V_{t_i}^{\text{hedge}} = \Delta_i S_{t_i} + B_i = C(S_{t_i}, t_i).$$

Between $t_i$ and $t_{i+1}$, the position is held; at $t_{i+1}$, update $\Delta_{i+1}$ and rebalance. The cash account evolves as $B_{i+1} = B_i e^{r\Delta t} - (\Delta_{i+1} - \Delta_i)S_{t_{i+1}}$ (buy/sell stock financed from cash).

The trader is *short* $Q = 20$ call contracts, each on $m = 100$ index units ($Qm = 2{,}000$ units). The initial premium inflow is

$$\Pi_0 = Q\,m\,C(S_0, 0) = \text{USD } 454{,}510.60.$$

She invests this premium to initiate the replicating (long-call) hedge. For each path,

$$\text{P\&L} = \big[\Pi_0 - Qm \cdot \text{Call Payoff at } T\big] \; + \; \big[\text{Terminal value of hedge at } T\big].$$

Under continuous rebalancing in Black–Scholes, P&L would be 0. With *weekly* rebalancing, there is discrete-hedging error.

**Simulation design.**

- Generate 10,000 GBM paths with 13 weekly steps.
- Along each path, compute $C(S_{t_i}, t_i)$ and $\Delta_i$ weekly, rebalance the hedge as above.
- At $T$, settle the short call and record the net P&L for the total position (short option + hedge).

**Results (10,000 paths, weekly rehedging).**

$$\begin{aligned} \text{Mean} &= -1{,}823.98 \text{ USD}, \\ \text{Std} &= 96{,}125.46 \text{ USD}, \\ P_1/P_5 &= -268{,}490.23 \ / \ -161{,}936.44 \text{ USD}, \\ \text{Median} &= 1{,}644.90 \text{ USD}, \\ P_{95}/P_{99} &= 150{,}711.98 \ / \ 223{,}821.70 \text{ USD}. \end{aligned}$$

A histogram of the P&L is approximately centered near zero with noticeable dispersion and moderately fat tails due to discrete hedging.

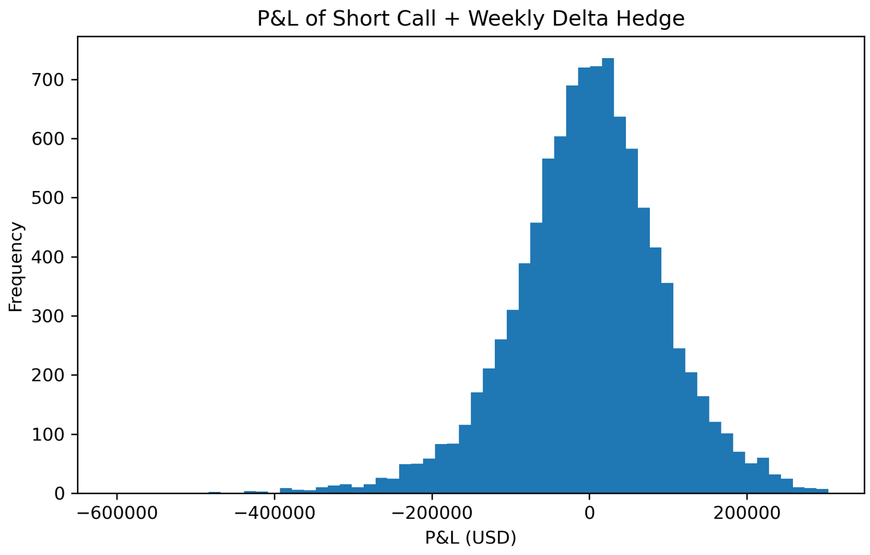

Figure 1: Histogram of P&L from dynamic delta hedging (weekly rebalancing)

**Interpretation**

- The mean P&L is close to zero (slightly negative), consistent with small *discrete-hedging error* when rebalancing only weekly rather than continuously.
- The large standard deviation reflects pathwise *gamma risk*: when the index moves sharply between re-hedges, the delta-only hedge cannot perfectly track the option's convexity, creating gains or losses.
- The median is slightly positive while the mean is slightly negative, indicating mild skew from convexity/theta interplay under discrete trading.
- Tails (e.g., $P_1$, $P_{99}$) quantify *gap risk*: infrequent (weekly) rebalancing leaves exposure to large moves between hedge dates.

## 2.c Dynamic Delta Hedging (Monthly) in Black–Scholes

We repeat the delta-hedging experiment from part (b), but now rebalance the hedge **monthly** instead of weekly, resulting in only three hedge adjustments over the 3-month horizon.

**Results (10,000 paths, monthly rehedging).**

$$\begin{aligned} \text{Mean} &= -2{,}694.21 \text{ USD}, \\ \text{Std} &= 190{,}595.65 \text{ USD}, \\ P_1/P_5 &= -549{,}533.94 \ / \ -336{,}748.33 \text{ USD}, \\ \text{Median} &= 8{,}276.66 \text{ USD}, \\ P_{95}/P_{99} &= 286{,}111.42 \ / \ 352{,}109.52 \text{ USD}. \end{aligned}$$

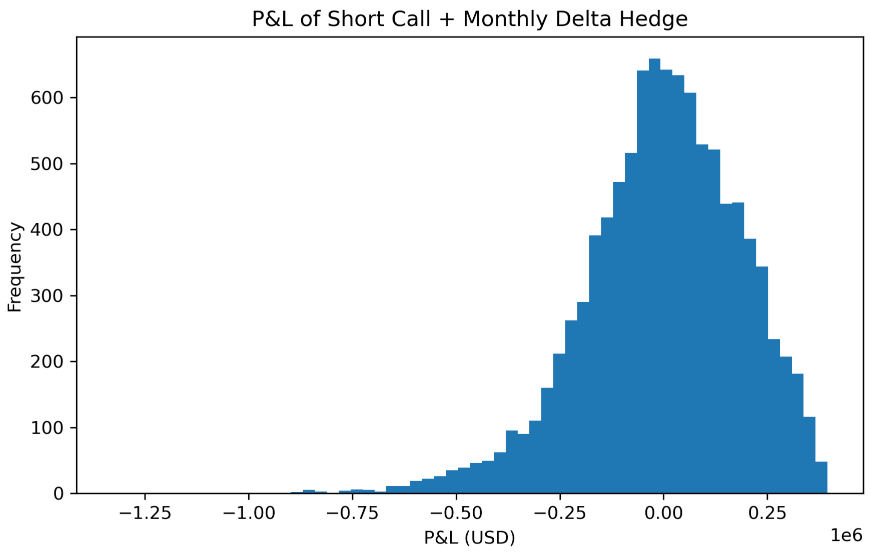

Figure 2: Histogram of P&L from dynamic delta hedging (monthly rebalancing)

**Interpretation** Compared with the weekly rehedging in part (b), the **mean loss increases** (from about −1,824 USD to −2,694 USD) and the **standard deviation roughly doubles**, reflecting higher residual risk. Less frequent rebalancing amplifies *discrete-hedging error* because the hedge cannot track fast market movements between adjustment dates. In essence, lower hedging frequency reduces transaction effort but significantly increases exposure to **gamma risk** and path-dependent P&L variation.

## 2.d Dynamic Delta Hedging (Daily) in Black–Scholes

We repeat the hedging experiment from part (b), but now adjust the hedge **daily** and **four times per day** to approximate continuous rebalancing in the Black–Scholes world.

**Results (10,000 paths, daily rehedging).**

$$\begin{aligned}
\text{Mean} &= -39.97 \text{ USD},\\
\text{Std} &= 44{,}305.43 \text{ USD},\\
P_1/P_5 &= -120{,}950.16 \ / \ -72{,}455.63 \text{ USD},\\
\text{Median} &= 463.46 \text{ USD},\\
P_{95}/P_{99} &= 70{,}539.70 \ / \ 114{,}972.76 \text{ USD}.
\end{aligned}$$

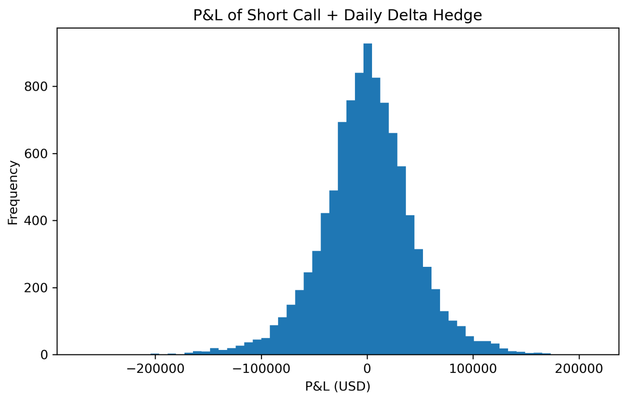

Figure 3: Histogram of P&L from dynamic delta hedging (daily rebalancing)

**Results (10,000 paths, four times daily rehedging).**

$$\begin{aligned}
\text{Mean} &= 227.21 \text{ USD},\\
\text{Std} &= 22{,}773.78 \text{ USD},\\
P_1/P_5 &= -61{,}221.84 \ / \ -36{,}667.87 \text{ USD},\\
\text{Median} &= 332.36 \text{ USD},\\
P_{95}/P_{99} &= 37{,}110.61 \ / \ 58{,}836.51 \text{ USD}.
\end{aligned}$$

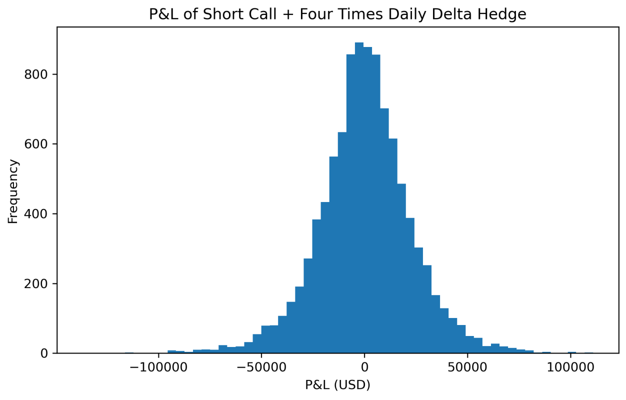

Figure 4: Histogram of P&L from dynamic delta hedging (four times daily rebalancing)

**Interpretation** Increasing the hedging frequency from weekly and monthly to daily substantially reduces both the bias and dispersion of the hedged portfolio's P&L distribution. The mean P&L is now close to zero and the variance has fallen sharply, indicating a more accurate replication of the option payoff. When rebalancing four times per day, the distribution tightens further, approaching the behavior expected under continuous trading in the Black–Scholes framework.

This convergence reflects the diminishing effect of **gamma risk**: as hedging intervals shorten, the delta position is adjusted more promptly to price movements, leaving less unhedged curvature exposure between rebalances. Consequently, frequent rehedging mitigates the nonlinearity losses arising from large intra-period price changes and drives the portfolio's performance closer to the theoretical, perfectly replicated outcome assumed in the model.

## 2.e Effect of the Drift Parameter on Hedging Performance

We repeat the daily delta-hedging experiment from part (d), but now assume higher expected returns for the underlying index: $\mu = 10\%$ and $\mu = 20\%$ instead of $\mu = 5\%$.

**Results (10,000 paths, daily rehedging, $\mu = 10\%$).**

$$\begin{aligned} \text{Mean} &= 262.57 \text{ USD},\\ \text{Std} &= 22{,}603.49 \text{ USD},\\ P_1/P_5 &= -60{,}989.92\ /\ -36{,}582.10 \text{ USD},\\ \text{Median} &= 401.71 \text{ USD},\\ P_{95}/P_{99} &= 36{,}515.97\ /\ 58{,}982.21 \text{ USD}. \end{aligned}$$

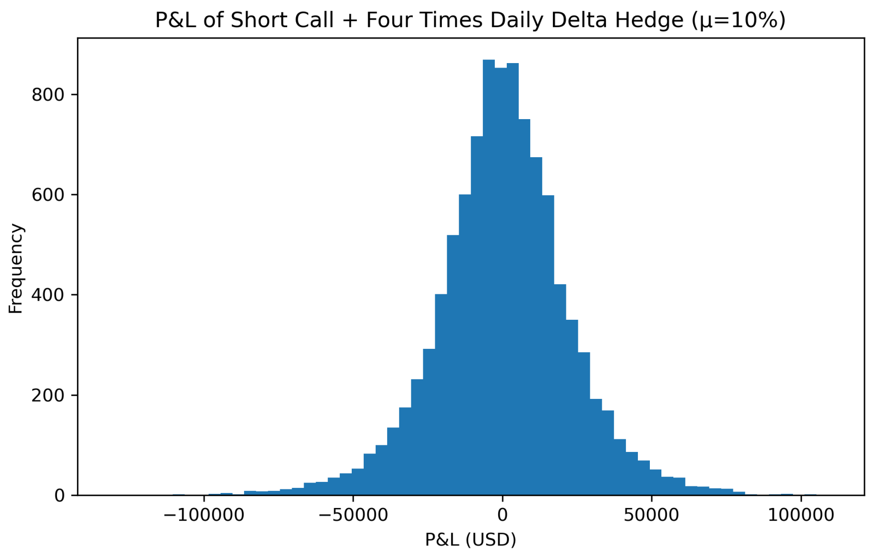

Figure 5: Histogram of P&L from dynamic delta hedging (four times daily rebalancing, $\mu = 10\%$)

**Results (10,000 paths, daily rehedging, $\mu = 20\%$).**

$$\begin{aligned} \text{Mean} &= 45.44 \text{ USD}, \\ \text{Std} &= 21{,}792.53 \text{ USD}, \\ P_1/P_5 &= -59{,}159.29 \ / \ -34{,}730.19 \text{ USD}, \\ \text{Median} &= -128.80 \text{ USD}, \\ P_{95}/P_{99} &= 35{,}242.10 \ / \ 57{,}180.97 \text{ USD}. \end{aligned}$$

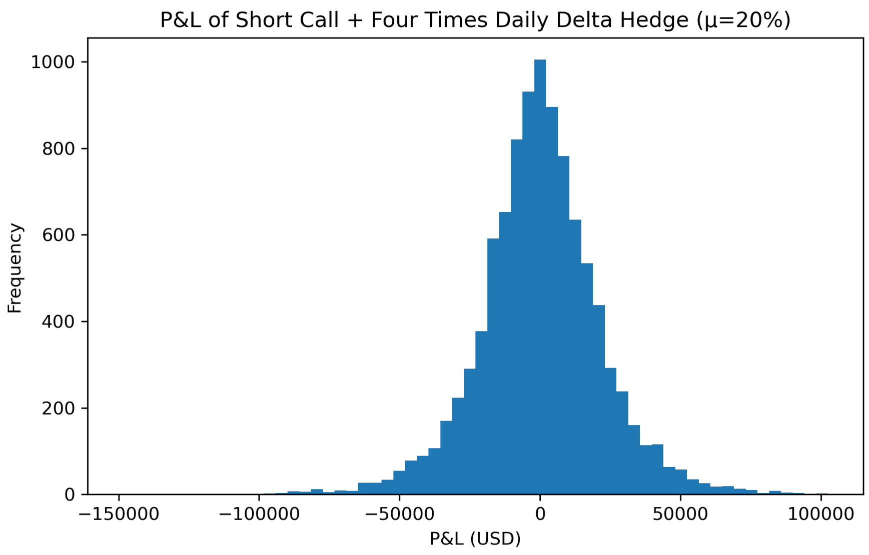

Figure 6: Histogram of P&L from dynamic delta hedging (four times daily rebalancing, $\mu = 20\%$)

**Interpretation** Increasing the drift parameter $\mu$ raises the expected growth rate of the underlying index, yet within the **risk-neutral Black–Scholes framework**, neither the option price nor the theoretical hedge ratios depend on $\mu$. Consequently, the hedge strategy-constructed under the assumption of risk-neutral dynamics-remains unchanged, while the realized P&L distribution reflects the higher upward drift of the simulated price paths.

When the true drift exceeds the risk-free rate, the trader's short call position becomes more likely to end up in-the-money, leading to slightly more negative hedging outcomes. This effect appears in the results: as $\mu$ increases from (5%) to (10%) and (20%), the mean P&L fluctuates around zero and the dispersion of outcomes remains largely similar, indicating that **drift mainly shifts the center of the distribution rather than its shape**. The hedge continues to perform well under daily rebalancing, but a higher physical drift introduces a mild systematic bias due to the deviation between the real-world and risk-neutral dynamics.

In summary, while the Black–Scholes hedge remains nearly perfect under frequent rebalancing, a persistent positive drift can slightly deteriorate the replication accuracy and bias the realized P&L toward small losses.

## 2.f The Role and Risk of Gamma Exposure

Gamma ($\Gamma$) measures the sensitivity of an option's delta ($\Delta$) to changes in the underlying price:

$$\Gamma = \frac{\partial^2 C}{\partial S^2}.$$

A trader with a **short gamma position** (as in the case of selling call options) faces the risk that delta changes rapidly when the underlying price moves. This means that even small price movements can cause large shifts in the hedge ratio, forcing frequent and costly rebalancing. When the market moves sharply up or down, a short-gamma trader must *buy high and sell low* to stay delta-neutral, leading to losses. Conversely, a long-gamma position benefits from volatility, as rebalancing tends to result in *buying low and selling high.*

Gamma risk becomes problematic because it introduces *nonlinear exposure* that cannot be neutralized by a static delta hedge. A trader short gamma suffers when realized volatility exceeds what was implied in the option's price. In practice, this means that while the position may appear hedged in small moves, large or sudden movements can generate significant losses due to unhedged curvature.

Traders are most concerned about their gamma exposure:

- **When volatility is high or unstable:** large, unpredictable moves in the underlying cause frequent delta adjustments and large P&L swings.
- **Near option expiration:** gamma increases sharply as time to maturity decreases, making the delta highly sensitive to small price changes.
- **Around key market events:** such as earnings announcements, policy decisions, or macroeconomic releases, when jump risk and volatility spikes are expected.

Gamma exposure determines how "curved" the option's value is with respect to the underlying. Short-gamma traders are effectively betting on market stability - if large price moves or volatility spikes occur, their hedges break down. Hence, managing gamma is critical to prevent large nonlinear losses, motivating delta-gamma hedging strategies that stabilize both first- and second-order sensitivities.

## 2.g Distribution of the Gamma position

Using a **daily hedging frequency** and **10,000 simulated paths**, I analysed the trader's *gamma exposure* and its relationship to the hedging performance.

The distribution of the **average gamma** across all paths (Figure 7) is strictly positive and centered roughly around $7 \times 10^{-4}$. This shape reflects that the option spends a substantial portion of the simulation close to being at-the-money, where gamma is highest, and less time deep in- or out-of-the-money, where gamma approaches zero.

Figure 8 plots the **absolute P&L** of the delta-hedged short-call position against the **average gamma** of each path. There is a clear upward-sloping relationship: paths with higher average gamma exhibit larger absolute hedging P&L. The computed correlation between average gamma and |P&L| is

$$\rho_{\Gamma,\,|\mathrm{P\&L}|} = 0.375,$$

confirming a moderate positive relationship between the curvature of the option value and the magnitude of hedging outcomes.

Gamma ($\Gamma$) measures the curvature of the option's price with respect to the underlying:

$$\Gamma = \frac{\partial^2 C}{\partial S^2}.$$

When gamma is large (near the strike), the delta changes rapidly as the underlying moves, making the hedge less effective between rebalancing times. Because the trader is **short the call option**, they are also **short gamma** and therefore lose money when the underlying moves sharply in either direction-they must continually “buy high and sell low” to maintain delta neutrality.

Hence, paths with higher average gamma correspond to greater curvature exposure and produce larger fluctuations in hedging P&L, explaining the positive empirical relationship.

**Mitigation of gamma risk.**

- **Increase hedging frequency:** rebalancing more frequently (e.g., intraday instead of daily) reduces the mismatch caused by rapid delta changes.
- **Delta-gamma hedging:** add another option position whose gamma offsets that of the short call, such that $\Gamma_{\text{portfolio}} = 0$.
- **Reduce exposure:** decrease position size during periods of high expected volatility or around major announcements.

The daily-hedged short-call position exhibits a positive dependence between gamma and the magnitude of P&L ($\rho = 0.375$). Higher gamma implies stronger nonlinearity and therefore greater potential hedging losses. Gamma risk can be reduced through more frequent rebalancing or through offsetting option positions.

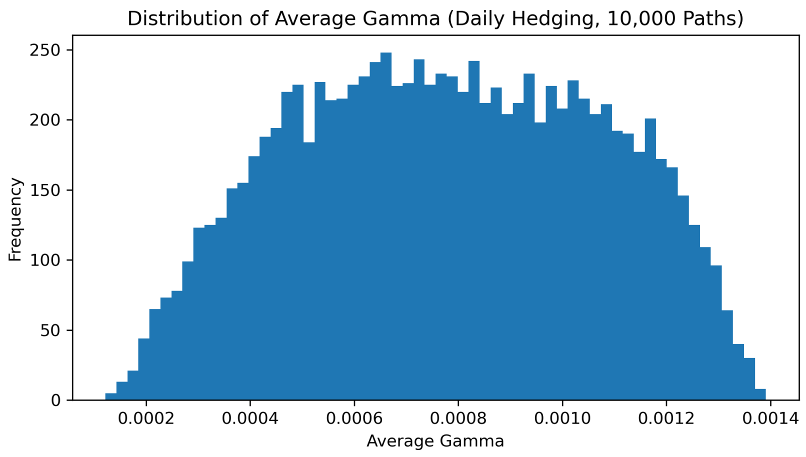

Figure 7: Distribution of Average Gamma (Daily Hedging, 10,000 Paths)

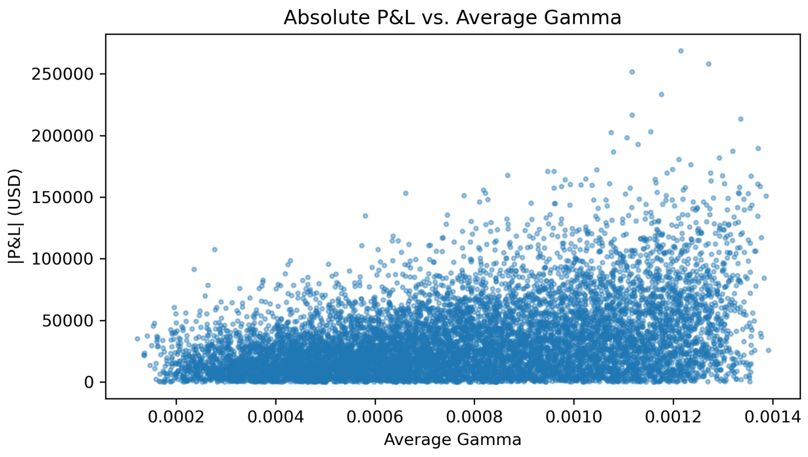

Figure 8: Absolute P&L vs. Average Gamma

## 3 Python Code

Listing 1: spf-assignment-1-script.py

```
# Import libraries -----
import pandas as pd
import numpy as np
import math

from scipy import stats
from scipy.stats import norm

import matplotlib.pyplot as plt


# Import data -----

# Read the CSV file into a DataFrame
dfSP500 = pd.read_csv("SP500.csv")

# Rename columns for consistency
dfSP500 = dfSP500.rename(columns={"SP500": "index_value"})

# Convert the observation_date column to datetime format
dfSP500['observation_date'] = pd.to_datetime(dfSP500['observation_date'])

# Calculate the number of months to cover in our sample
iNumber_of_months = 10*(2 + 11) - 2*11

# Select the last iNumber_of_months observations and reset the index
dfSP500 = dfSP500[len(dfSP500)-iNumber_of_months:].reset_index(drop=True)

# Calculating simple net returns using the definition -----
for i in range(iNumber_of_months-1):
```

```python
    dfSP500.loc[i, "simple_net_return"] = (dfSP500.loc[i+1, "index_value"]
       / dfSP500.loc[i, "index_value"]) - 1

# Define sample size
iSample_size = len(dfSP500) - 1

# Solution to (a) -----

# Parameters
S0 = dfSP500["index_value"].iloc[-1]
K = 6500
T = 0.25
r = 0.03
sigma = np.std(dfSP500["simple_net_return"], ddof=1) * np.sqrt(12)  #
    Annualized volatility
N = 300

# Time step and factors
dt = T / N
u = np.exp(sigma * np.sqrt(dt))
d = 1/u
p = (np.exp(r * dt) - d) / (u - d)

# Terminal stock prices
ST = S0 * u**np.arange(N, -1, -1) * d**np.arange(0, N+1)

# Terminal payoffs
C = np.maximum(ST - K, 0)

# Backward induction
for _ in range(N):
    C = np.exp(-r * dt) * ((1-p) * C[1:] + (p) * C[:-1])

C_binomial = C[0]

#  Black-Scholes price
d1 = (np.log(S0 / K) + (r + 0.5 * sigma**2) * T) / (sigma * np.sqrt(T))
d2 = d1 - sigma * np.sqrt(T)
C_bs = S0 * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2)

print(f"Binomial tree price (300 steps): {C_binomial:.4f}")
print(f"Black-Scholes price: {C_bs:.4f}")
print(f"Difference: {abs(C_binomial - C_bs):.6f}")

# Solution to (b) -----

# Parameters
r_annual = np.log(1 + r)  # continuously compounded
T_months = 3
dt = 1  # month
```

```python
steps = int(T_months / dt)
r_month = r_annual / 12
sigma_monthly = sigma / np.sqrt(12)

# Monte Carlo simulation using Euler discretization
def monte_carlo_euler(M):
    payoffs = np.zeros(M)
    for m in range(M):
        S = S0
        for _ in range(steps):
            phi = np.random.randn()  # standard normal
            S = S + S * (r_month + sigma_monthly * np.sqrt(dt) * phi)
        payoffs[m] = max(S - K, 0)
    return np.exp(-r_annual * T) * np.mean(payoffs)

# Simulate for different M
for M in [100, 500, 1000, 5000, 10000]:
    price = monte_carlo_euler(M)
    print(f"M = {M:5d} : Option price = {price:.4f}")


# Solution to (c) -----

N = 63             # trading days
dt = T / N         # one-day step in years

def monte_carlo_daily(M):
    payoffs = np.zeros(M)
    for m in range(M):
        S = S0
        for _ in range(N):
            phi = np.random.randn()
            S = S + S * (r_annual * dt + sigma * np.sqrt(dt) * phi)
        payoffs[m] = max(S - K, 0)
    return np.exp(-r_annual * T) * np.mean(payoffs)

# --- Simulate for different M ---
for M in [100, 500, 1000, 5000, 10000]:
    price = monte_carlo_daily(M)
    print(f"M = {M:5d} : Option price = {price:.4f}")



# Solution to (d) -----

def monte_carlo_logS(M):
    payoffs = np.zeros(M)
    logS0 = np.log(S0)
    for m in range(M):
        logS = logS0
        for _ in range(N):
            phi = np.random.randn()
            logS += (r - 0.5 * sigma**2) * dt + sigma * np.sqrt(dt) * phi
        ST = np.exp(logS)
        payoffs[m] = max(ST - K, 0)
    return np.exp(-r * T) * np.mean(payoffs)

# --- Simulate for different M ---
for M in [100, 500, 1000, 5000, 10000]:
    price = monte_carlo_logS(M)
    print(f"M = {M:5d} : Option price = {price:.4f}")


# Solution to (e) -----

def monte_carlo_exact(M):
    Z = np.random.randn(M)
    ST = S0 * np.exp((r - 0.5 * sigma**2) * T + sigma * np.sqrt(T) * Z)
    payoffs = np.maximum(ST - K, 0)
    return np.exp(-r * T) * np.mean(payoffs)

# --- Simulate for different M ---
for M in [100, 500, 1000, 5000, 10000]:
    price = monte_carlo_exact(M)
    print(f"M = {M:5d} : Option price = {price:.4f}")


# Solution to (f) -----

M = 10000
K1 = 6600
K2 = 6500
dt = T / N

# Simulation using log Euler
logS0 = np.log(S0)
payoffs = np.zeros(M)

for m in range(M):
    logS = logS0
    for _ in range(N):
        phi = np.random.randn()
        logS += (r_annual - 0.5 * sigma**2) * dt + sigma * np.sqrt(dt) * phi
    ST = np.exp(logS)
    payoffs[m] = np.where(ST > K2, ST - K1, 0)


# Discounted expected payoff
gap_call_price = np.exp(-r_annual * T) * np.mean(payoffs)
print(f"Gap call option price = {gap_call_price:.2f} USD")
```

```


# Solution to (a) -----

# Parameters
S0 = 6500
K = 6500
contract_size = 100
num_contracts = 20

# Black-Scholes d1 and d2
d1 = (np.log(S0 / K) + (r_annual + 0.5 * sigma**2) * T) / (sigma *
    np.sqrt(T))
d2 = d1 - sigma * np.sqrt(T)

# Black-Scholes call price per unit
call_price = S0 * norm.cdf(d1) - K * np.exp(-r_annual * T) * norm.cdf(d2)

# Total cash received
total_cash = call_price * contract_size * num_contracts

# Display results
print(f"Call price per index unit: {call_price:.4f} USD")
print(f"Total cash received (20Œ100 contracts): {total_cash:,.2f} USD")

# Solution to (b) -----

# Black-Scholes call price & delta
def bs_call_price_delta(S, K, r, sigma, tau):
    """Return (price, delta) of a European call (continuous r)."""
    if tau <= 0:
        price = max(S - K, 0.0)
        delta = float(S > K)
        return price, delta
    sqrt_tau = np.sqrt(tau)
    d1 = (np.log(S / K) + (r + 0.5 * sigma**2) * tau) / (sigma * sqrt_tau)
    d2 = d1 - sigma * sqrt_tau
    price = S * stats.norm.cdf(d1) - K * np.exp(-r * tau) *
        stats.norm.cdf(d2)
    delta = stats.norm.cdf(d1)
    return price, delta

# Simulate GBM paths
def simulate_paths(S0, mu, sigma, T, steps, M, seed=42):
    """Simulate GBM under P, shape (M, steps+1)."""
    rng = np.random.default_rng(seed)
    dt = T / steps
    S = np.empty((M, steps + 1))
    S[:, 0] = S0
    for n in range(steps):
```

```python
        Z = rng.standard_normal(M)
        S[:, n + 1] = S[:, n] * np.exp((mu - 0.5 * sigma**2) * dt + sigma *
            np.sqrt(dt) * Z)
    return S

# Perform delta-hedging simulation
def delta_hedge_sim(S_paths, K, r, sigma, taus, units):
    """Simulate delta-hedged P&L for short call."""
    M, steps_plus1 = S_paths.shape
    steps = steps_plus1 - 1
    dt = taus[0] - taus[1]  # uniform spacing
    pnl = np.zeros(M)

    for m in range(M):
        S_series = S_paths[m, :]
        C0, delta0 = bs_call_price_delta(S_series[0], K, r, sigma, taus[0])
        cash = C0 * units                         # receive option premium
        shares = delta0 * units
        cash -= shares * S_series[0]              # finance shares

        # Rehedge at each step (excluding maturity)
        for n in range(1, steps):
            cash *= math.exp(r * dt)              # accrue interest
            Cn, deltan = bs_call_price_delta(S_series[n], K, r, sigma,
                taus[n])
            dshares = deltan * units - shares
            cash -= dshares * S_series[n]
            shares += dshares

        # Final step: accrue, close hedge, pay payoff
        cash *= math.exp(r * dt)
        ST = S_series[-1]
        cash += shares * ST
        payoff = max(ST - K, 0.0) * units
        pnl[m] = cash - payoff

    return pnl

# Delta-hedging with parameters
def run_delta_hedge(mu=0.05, sigma_monthly=sigma_monthly, steps=13, T=0.25,
    M=10000):
    """Run full simulation and print results."""
    # Parameters
    S0, K = 6500.0, 6500.0
    r_nominal = 0.03
    r = np.log(1.0 + r_nominal / 4.0) / 0.25
    sigma = sigma_monthly * np.sqrt(12.0)
    contract_size, num_contracts = 100, 20
    units = contract_size * num_contracts

    # Setup
    times = np.linspace(0, T, steps + 1)
    taus = T - times

    # Simulate & hedge
    S_paths = simulate_paths(S0, mu, sigma, T, steps, M)
    pnl = delta_hedge_sim(S_paths, K, r, sigma, taus, units)

    # Summary
    stats_dict = {
        "Mean": np.mean(pnl),
        "Std": np.std(pnl, ddof=1),
        "P1": np.percentile(pnl, 1),
        "P5": np.percentile(pnl, 5),
        "Median": np.percentile(pnl, 50),
        "P95": np.percentile(pnl, 95),
        "P99": np.percentile(pnl, 99),
    }

    return pnl, stats_dict



# Run delta-hedging simulation
pnl, stats_dict = run_delta_hedge()

print(f"Delta-hedged P&L over {T*12:.0f} weeks (weekly rehedging, {M:,}
    paths):")
for k, v in stats_dict.items():
    print(f"{k:6}: {v:,.2f} USD")

# Plot
plt.figure(figsize=(7, 4.5))
plt.hist(pnl, bins=60)
plt.title("P&L of Short Call + Weekly Delta Hedge")
plt.xlabel("P&L (USD)")
plt.ylabel("Frequency")
plt.tight_layout()
plt.savefig("Documentation/Figures/delta_hedge_pnl_weekly.png", dpi=300)
plt.show()

# Solution to (c) -----

# Run delta-hedging simulation
# Now with monthly hedging (steps=3)
pnl, stats_dict = run_delta_hedge(steps=3)

print(f"Delta-hedged P&L over {T*12:.0f} weeks (monthly rehedging, {M:,}
    paths):")
for k, v in stats_dict.items():
    print(f"{k:6}: {v:,.2f} USD")

# Plot
plt.figure(figsize=(7, 4.5))
plt.hist(pnl, bins=60)
plt.title("P&L of Short Call + Monthly Delta Hedge")
plt.xlabel("P&L (USD)")
plt.ylabel("Frequency")
plt.tight_layout()
plt.savefig("Documentation/Figures/delta_hedge_pnl_monthly.png", dpi=300)
plt.show()

# Solution to (d) part 1 -----

# Run delta-hedging simulation
# Now with daily hedging (steps=63)
pnl, stats_dict = run_delta_hedge(steps=63)

print(f"Delta-hedged P&L over {T*12:.0f} weeks (daily rehedging, {M:,}
    paths):")
for k, v in stats_dict.items():
    print(f"{k:6}: {v:,.2f} USD")

# Plot
plt.figure(figsize=(7, 4.5))
plt.hist(pnl, bins=60)
plt.title("P&L of Short Call + Daily Delta Hedge")
plt.xlabel("P&L (USD)")
plt.ylabel("Frequency")
plt.tight_layout()
plt.savefig("Documentation/Figures/delta_hedge_pnl_daily.png", dpi=300)
plt.show()

# Solution to (d) part 2 -----

# Run delta-hedging simulation
# Now with four times daily hedging (steps=63*4=252)
pnl, stats_dict = run_delta_hedge(steps=63*4)

print(f"Delta-hedged P&L over {T*12:.0f} weeks (four times daily rehedging,
    {M:,} paths):")
for k, v in stats_dict.items():
    print(f"{k:6}: {v:,.2f} USD")

# Plot
plt.figure(figsize=(7, 4.5))
plt.hist(pnl, bins=60)
plt.title("P&L of Short Call + Four Times Daily Delta Hedge")
plt.xlabel("P&L (USD)")
plt.ylabel("Frequency")
plt.tight_layout()
plt.savefig("Documentation/Figures/delta_hedge_pnl_four_times_daily.png",
    dpi=300)
plt.show()

# Solution to (e) part 1 -----

# Run delta-hedging simulation
# Now with four times daily hedging (steps=63*4=252) and mu=0.10
pnl, stats_dict = run_delta_hedge(steps=63*4, mu=0.10)

print(f"Delta-hedged P&L over {T*12:.0f} weeks (four times daily rehedging,
    {M:,} paths, mu=10%):")
for k, v in stats_dict.items():
     print(f"{k:6}: {v:,.2f} USD")

# Plot
plt.figure(figsize=(7, 4.5))
plt.hist(pnl, bins=60)
plt.title("P&L of Short Call + Four Times Daily Delta Hedge (mu=10%)")
plt.xlabel("P&L (USD)")
plt.ylabel("Frequency")
plt.tight_layout()
plt.savefig("Documentation/Figures/delta_hedge_pnl_four_times_daily_mu_10.png",
    dpi=300)
plt.show()

# Solution to (e) part 2 -----

# Run delta-hedging simulation
# Now with four times daily hedging (steps=63*4=252) and mu=0.20
pnl, stats_dict = run_delta_hedge(steps=63*4, mu=0.20)

print(f"Delta-hedged P&L over {T*12:.0f} weeks (four times daily rehedging,
    {M:,} paths, mu=20%):")
for k, v in stats_dict.items():
     print(f"{k:6}: {v:,.2f} USD")

# Plot
plt.figure(figsize=(7, 4.5))
plt.hist(pnl, bins=60)
plt.title("P&L of Short Call + Four Times Daily Delta Hedge (mu=20%)")
plt.xlabel("P&L (USD)")
plt.ylabel("Frequency")
plt.tight_layout()
plt.savefig("Documentation/Figures/delta_hedge_pnl_four_times_daily_mu_20.png",
    dpi=300)
plt.show()

# Solution to (g) -----

# Black-Scholes gamma
def bs_call_gamma(S, K, r, sigma, tau):
     """Return the Black-Scholes gamma."""
     if tau <= 0:
         return 0.0
     sqrt_tau = np.sqrt(tau)
     d1 = (np.log(S / K) + (r + 0.5 * sigma**2) * tau) / (sigma * sqrt_tau)
     gamma = stats.norm.pdf(d1) / (S * sigma * sqrt_tau)
     return gamma

# Black-Scholes call price, delta, and gamma
def bs_call_price_delta_gamma(S, K, r, sigma, tau):
     """Return (price, delta, gamma) for European call."""
     if tau <= 0:
         payoff = max(S - K, 0.0)
         return payoff, float(S > K), 0.0
     sqrt_tau = np.sqrt(tau)
     d1 = (np.log(S / K) + (r + 0.5 * sigma**2) * tau) / (sigma * sqrt_tau)
     d2 = d1 - sigma * sqrt_tau
     price = S * stats.norm.cdf(d1) - K * np.exp(-r * tau) *
         stats.norm.cdf(d2)
     delta = stats.norm.cdf(d1)
     gamma = stats.norm.pdf(d1) / (S * sigma * sqrt_tau)
     return price, delta, gamma

# Perform delta-hedging simulation with gamma tracking
def delta_hedge_with_gamma(S_paths, K, r, sigma, taus, units):
     M, steps_plus1 = S_paths.shape
     steps = steps_plus1 - 1
     dt = taus[0] - taus[1]
     pnl = np.zeros(M)
     avg_gamma = np.zeros(M)

     for m in range(M):
         S_series = S_paths[m, :]
         C0, delta0, gamma0 = bs_call_price_delta_gamma(S_series[0], K, r,
             sigma, taus[0])
         cash = C0 * units
         shares = delta0 * units
         cash -= shares * S_series[0]

         gammas = [gamma0]

         for n in range(1, steps):
             cash *= math.exp(r * dt)
             Cn, deltan, gamman = bs_call_price_delta_gamma(S_series[n], K,
                 r, sigma, taus[n])
             dshares = deltan * units - shares
             cash -= dshares * S_series[n]
             shares += dshares
             gammas.append(gamman)

         # Final step
         cash *= math.exp(r * dt)
         ST = S_series[-1]
         cash += shares * ST
         payoff = max(ST - K, 0.0) * units
         pnl[m] = cash - payoff
         avg_gamma[m] = np.mean(gammas)

     return pnl, avg_gamma


 # Run the experiment (daily hedging)
 S0 = 6500.0
 K = 6500.0
 T = 0.25
 steps = 63
 M = 10000
 mu = 0.05
 r_nominal = 0.03
 r = np.log(1 + r_nominal / 4) / 0.25
 sigma = sigma_monthly * np.sqrt(12)
 contract_size = 100
 num_contracts = 20
 units = contract_size * num_contracts

 times = np.linspace(0, T, steps + 1)
 taus = T - times

 S_paths = simulate_paths(S0, mu, sigma, T, steps, M)
 pnl, avg_gamma = delta_hedge_with_gamma(S_paths, K, r, sigma, taus, units)


 # Analysis & plots
 # 1. Distribution of gamma
 plt.figure(figsize=(7,4))
 plt.hist(avg_gamma, bins=60)
 plt.title("Distribution of Average Gamma (Daily Hedging, 10,000 Paths)")
 plt.xlabel("Average Gamma")
 plt.ylabel("Frequency")
 plt.tight_layout()
 plt.savefig("Documentation/Figures/avg_gamma_distribution.png", dpi=300)
 plt.show()

 # 2. Relationship between avg gamma and |PnL|
 abs_pnl = np.abs(pnl)
 plt.figure(figsize=(7,4))
 plt.scatter(avg_gamma, abs_pnl, s=6, alpha=0.4)
 plt.title("Absolute P&L vs. Average Gamma")
 plt.xlabel("Average Gamma")
 plt.ylabel("|P&L| (USD)")
 plt.tight_layout()
 plt.savefig("Documentation/Figures/abs_pnl_vs_avg_gamma.png", dpi=300)
 plt.show()

 # 3. Correlation
 corr = np.corrcoef(avg_gamma, abs_pnl)[0, 1]
 print(f"Correlation between average gamma and |PnL|: {corr:.3f}")
```