Indian Institute of Technology Tirupati

Department of Mathematics and Statistics

Delta Hedging Simulation and Gamma Risk Analysis

DEBASISH PRADHAN

ROLL NO: MA24M004

MSc Mathematics & Statistics

IIT Tirupati

Subject: Statistical Finance

Academic Year: 2025–2026

CONTENTS

Introduction

This project examines the behaviour of delta hedging and gamma risk for an option based on AAPL stock. Delta hedging reduces the portfolio's sensitivity to small changes in the underlying stock price, while gamma represents how quickly delta itself changes. Using real market data, the study observes hedge adjustments, P&L movements, and how risk evolves as the option approaches expiration.

Objectives

The objective of this project is to analyze the performance of daily delta hedging and evaluate the impact of gamma on hedging efficiency. The study focuses on hedge error, volatility behaviour, and the overall effectiveness of the hedged portfolio.

Data Collection

The dataset used in this project contains historical daily AAPL stock prices. The data includes the following columns: Date, Open, High, Low, Close, Adjusted Close, and Volume. These fields form the foundation for calculating delta, gamma, and other relevant risk metrics required for the simulation.

The dataset covers the period from **2010-08-02** to **2020-07-30**, providing a long and consistent time range suitable for analysing market behaviour and hedging performance. The data was obtained from a publicly available Kaggle dataset.

Data Source: https://www.kaggle.com/datasets/tarunpaparaju/apple-aapl-historical-stock-data

Terminology

Delta: Delta measures the sensitivity of an option's price to small movements in the underlying stock price. A delta-neutral position reduces this sensitivity.

Gamma: Gamma measures the rate at which delta changes. Higher gamma means delta adjusts more rapidly as the stock price moves.

Hedge Ratio: The number of shares required to hedge the delta of an option. A delta hedge typically holds $-\Delta$ shares per option.

P&L (Profit and Loss): Tracks the daily or cumulative gain/loss of the hedged portfolio during the simulation.

Rebalancing: Adjusting the number of shares held in the hedge as delta changes over time.

Volatility: The degree of variation in stock price. It affects both delta and gamma values through the Black–Scholes model.

Time to Expiration: The remaining time until the option expires. Gamma typically increases as expiration approaches.

Underlying Asset: The stock on which the option is written; in this case, AAPL.

DELTA HEDGING AND GAMMA ANALYSIS

To achieve objective of this project, we combine mathematical modelling with simulation techniques. Delta and gamma, the two most important first- and second-order sensitivities of an option, allow us to understand how the option reacts to movements in the stock price.

Mathematically, delta represents the **rate of change of the option price with respect to the stock price**, while gamma represents the **rate of change of delta itself**. Together, these two quantities tell us how stable or unstable a hedged portfolio will be as the market evolves. Our general approach simulates daily adjustments in the hedge and evaluates how both delta and gamma influence the final profit, loss, and stability of the portfolio.

(a) GENERAL APPROACH

The simulation follows a structured sequence:

- compute the option's delta and gamma every day,
- adjust the hedge based on the new delta,
- update the portfolio value,
- record the daily P&L,
- repeat until expiration.

This framework allows us to observe how delta hedging behaves dynamically and how gamma affects the accuracy of the hedge as the stock price moves.

Why this helps achieve the objective: This approach gives a day-by-day view of hedge stability. If small stock movements produce large changes in portfolio value, the hedge is unstable — usually due to high gamma. A good hedging strategy should produce a smooth, low-volatility portfolio value path, and this methodology measures exactly that.

Pseudocode (Overall Workflow):

```
for each trading day t:
    compute delta(t)
    compute gamma(t)
    adjust hedge -> hedge_shares(t) = -delta(t)
    compute portfolio value V(t)
    compute daily PnL
end for
```

i. Delta Calculation

Delta measures how much the option price changes for a small change in the stock price:

$$\Delta = N(d_1)$$

where

$$d_1 = \frac{\ln(S/K) + (r + \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}}.$$

Intuition: If one option behaves like 0.55 shares of stock, then holding -0.55 shares neutralizes the directional risk. Delta hedging aims to make the portfolio delta-neutral so that small price changes in the stock have minimal effect.

Why delta helps in this project: Our objective is to minimize hedge error and volatility. Delta is the key tool for neutralizing small price movements. Without delta adjustments, the portfolio would be fully exposed to stock fluctuations.

Pseudocode:

```
function compute_delta(S, K, T, r, sigma):
    d1 = (log(S/K) + (r + 0.5*sigma^2)*T) / (sigma*sqrt(T))
    return N(d1)
```

ii. Gamma Calculation

Gamma measures the curvature of the option price curve and how quickly delta changes:

$$\Gamma = \frac{N'(d_1)}{S\sigma\sqrt{T}}$$

Intuition: High gamma means delta changes rapidly. Even a small stock price change forces a large hedge adjustment. This is why hedging becomes more difficult near expiration and when the option is at-the-money.

Why gamma helps in this project: Gamma explains the source of hedge error. A strategy that considers only delta but ignores gamma will frequently under- or over-hedge. Understanding gamma helps us interpret spikes in P&L volatility as expiration approaches.

Pseudocode:

```
function compute_gamma(S, K, T, r, sigma):
    d1 = ...
    return n(d1) / (S * sigma * sqrt(T))
```

iii. Daily Rebalancing

Every day, the hedge must be updated to match the new delta:

$$HedgePosition(t) = -\Delta(t)$$

Intuition: The hedge must “follow” the option. When the stock price moves, delta changes; therefore, the number of shares required to stay neutral also changes.

Why this helps the objective: Daily rebalancing prevents large deviations of the portfolio value caused by stock movement. Without this step, hedge error would accumulate quickly.

Pseudocode:

```
hedge_shares[t] = -delta[t]
portfolio_value[t] = C[t] + hedge_shares[t] * S[t]
```

iv. Portfolio Tracking

The value of the hedged portfolio is:

$$V_t = C_t - \Delta_t S_t$$

Intuition: A perfectly hedged portfolio should look flat over time. Any movement in V_t suggests instability in the hedge — often caused by gamma.

Why this supports the objective: Our goal is to study hedge efficiency. Tracking V_t is the direct way to measure how well the strategy works.

Pseudocode:

```
pnl[t] = V[t] - V[t-1]
track all V[t] and pnl[t]
```

v. Performance Metrics

We evaluate hedge quality using:

$$PnL = V_{t+1} - V_t$$

$$PnLVolatility = \sqrt{\mathrm{Var}(PnL)}$$

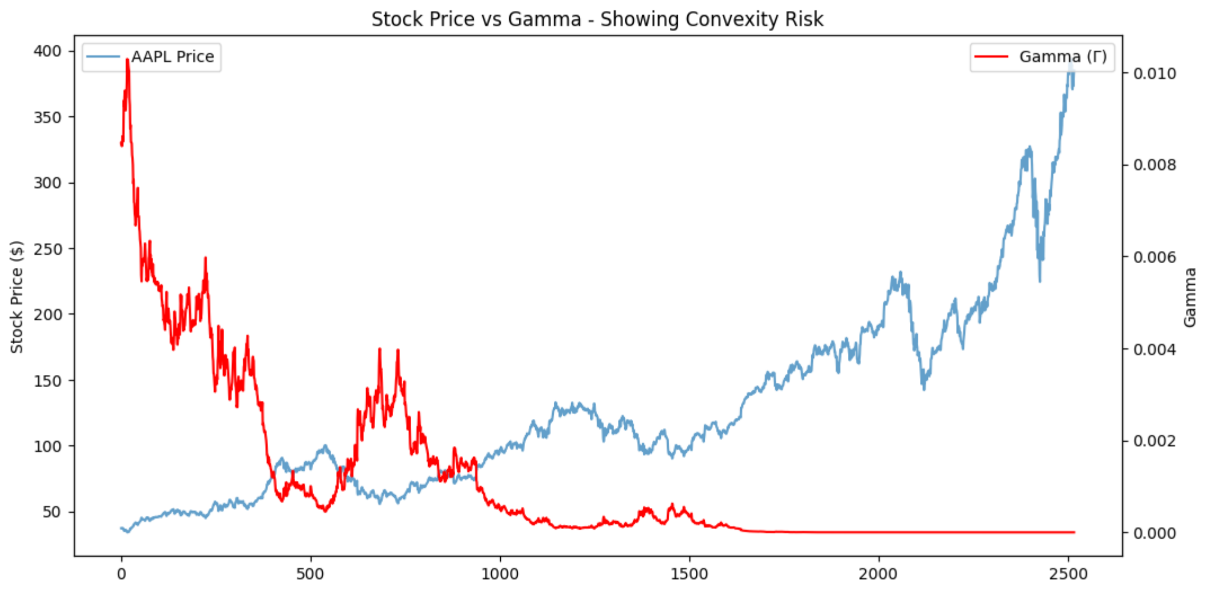
$$HedgeError = V_{final} - V_{initial}$$

Why these metrics matter: Delta hedging aims for minimal volatility and near-zero hedge error. Large PnL swings indicate gamma-driven instability.

SIMULATION OUTPUT AND INTERPRETATION

To evaluate the behaviour of delta hedging and gamma risk, we generated two important visual outputs. These graphs help us understand how the stock moves, how gamma behaves, and how the hedged portfolio reacts under real market data. The interpretation below explains in simple terms how each graph reflects the risk and performance of the hedging strategy.

1. Stock Price vs Gamma (Convexity Risk)



What this graph shows: The blue curve represents the AAPL stock price over time. The red curve represents the gamma of the option.

Gamma is highest when:

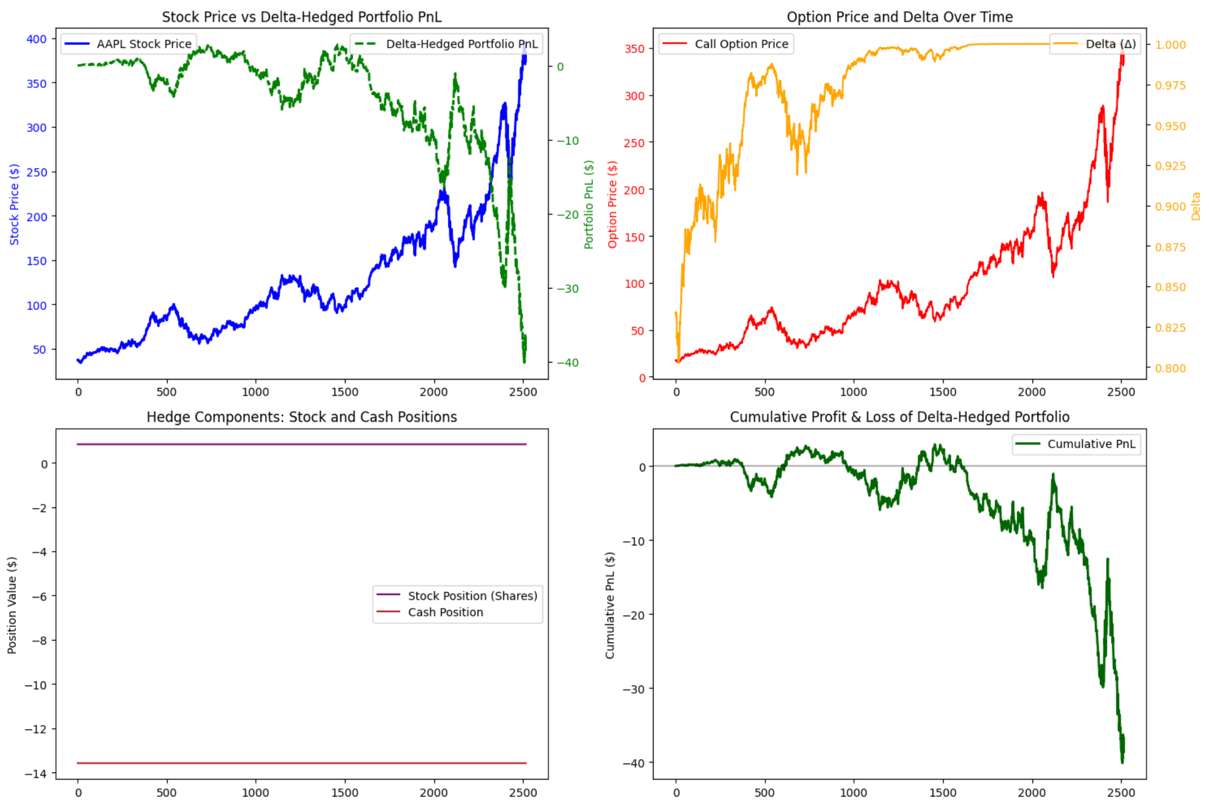
- the option is at-the-money (ATM),
- the stock is very volatile,
- time to expiration is short.

Intuition for a child: Think of delta as a steering wheel. Gamma tells you **how quickly the steering wheel turns**. If gamma is high, even a small movement makes a big change.

Why this helps our objective: Our aim is to measure hedge effectiveness. Gamma explains hedge instability. High gamma means delta changes very fast → we must rebalance often → hedge becomes costly and imperfect.

What we learn: Early in the dataset, gamma spikes significantly — this is where the hedge is most unstable. As time moves forward, gamma gradually falls close to zero, making the hedge more stable.

2. Combined Simulation Output (Four-Panel Figure)



This combined figure contains **four essential panels** which together explain how the delta hedge and portfolio evolve. Each sub-panel is interpreted below.

(a) Stock Price vs Delta-Hedged Portfolio P&L

The green line shows portfolio P&L, while the blue line shows the AAPL price. A perfectly hedged portfolio should remain flat, but here we see fluctuations.

Why these fluctuations appear:

- Gamma causes delta to change rapidly.
- We hedge in discrete time, not continuous time.
- Sudden stock jumps cannot be hedged instantly.

Conclusion: The hedge works reasonably for stable periods but breaks when gamma is high.

(b) Option Price and Delta Over Time

The option price rises with the stock, and delta approaches 1 as the option moves deep in-the-money.

Meaning: When delta = 1, the option behaves just like the stock. Hedging becomes expensive because we must short almost one share per option.

Intuition: When the stock goes up a lot, the option "acts like the stock," so delta increases.

(c) Hedge Components: Stock and Cash Positions

This panel shows how many shares we hold (purple) and how much cash we use (brown).

Why useful: It reveals how much we must adjust the hedge every day. Large movements in stock or cash show unstable delta.

(d) Cumulative P&L of the Delta-Hedged Portfolio

This graph shows the total profit or loss as time progresses.

What we see:

- Long flat regions → stable delta periods.
- Occasional sharp drops → gamma spikes causing hedge error.
- Final drop → accumulated hedging error at the end.

Key insight: Perfect hedging is impossible in real markets because trading is discrete and gamma forces constant rebalancing.

PERFORMANCE METRICS

Metric	Value	Explanation
Total PnL	-38.63	Total profit or loss from the hedged strategy. Negative means imperfect hedging.
PnL Volatility	0.4723	Measures how stable the hedged portfolio is. Higher value = unstable hedge.
Max PnL	2.9272	Best profit observed in the simulation.
Min PnL	-40.1424	Largest loss, mostly caused by gamma spikes.
Final Hedge Error	-38.63	Final net deviation. Ideally 0 for a perfect hedge.
Stock Volatility	0.0173	Daily volatility of AAPL stock.
Portfolio Volatility	0.0918	Volatility of hedged portfolio; shows remaining risk.

Interpretation: The negative total PnL and hedge error mean the hedge was imperfect—exactly what we expect from discrete hedging under high gamma. PnL volatility highlights unstable periods caused by stock jumps or rapid delta changes.

SUMMARY OF WHAT WE ACHIEVED

1. Visualized how delta and gamma drive hedge behaviour.

2. Observed that high gamma leads to unstable hedging and larger P&L swings.

3. Studied the hedge components (stock + cash) and saw how they evolve.

4. Quantified risk using performance metrics and saw that:

- hedge error = -38.63 (imperfect hedge),
- portfolio volatility > stock volatility,
- P&L becomes unstable during gamma spikes.

5. Most importantly, we learned: *Discrete delta hedging cannot perfectly replicate continuous hedging. Gamma risk guarantees some level of hedge error.*

This is the key lesson of the project.

APPENDIX–I (PYTHON CODE)

Below is the complete Python notebook used for the delta-hedging simulation. The notebook contains data loading, Black–Scholes Greeks (delta gamma), the daily rebalancing simulation, performance metric calculations, and all plotting routines. The notebook pages are included verbatim for reference.

```
In [1]: import pandas as pd
        import numpy as np
        import yfinance as yf
        from scipy.stats import norm
        import matplotlib.pyplot as plt
```

```
In [2]: aapl_data = pd.read_csv('AAPL.csv')
```

```
In [3]: print(f"Apple data shape: {aapl_data.shape}")
        print(f"Date range: {aapl_data.index[0]} to {aapl_data.index[-1]}")
```

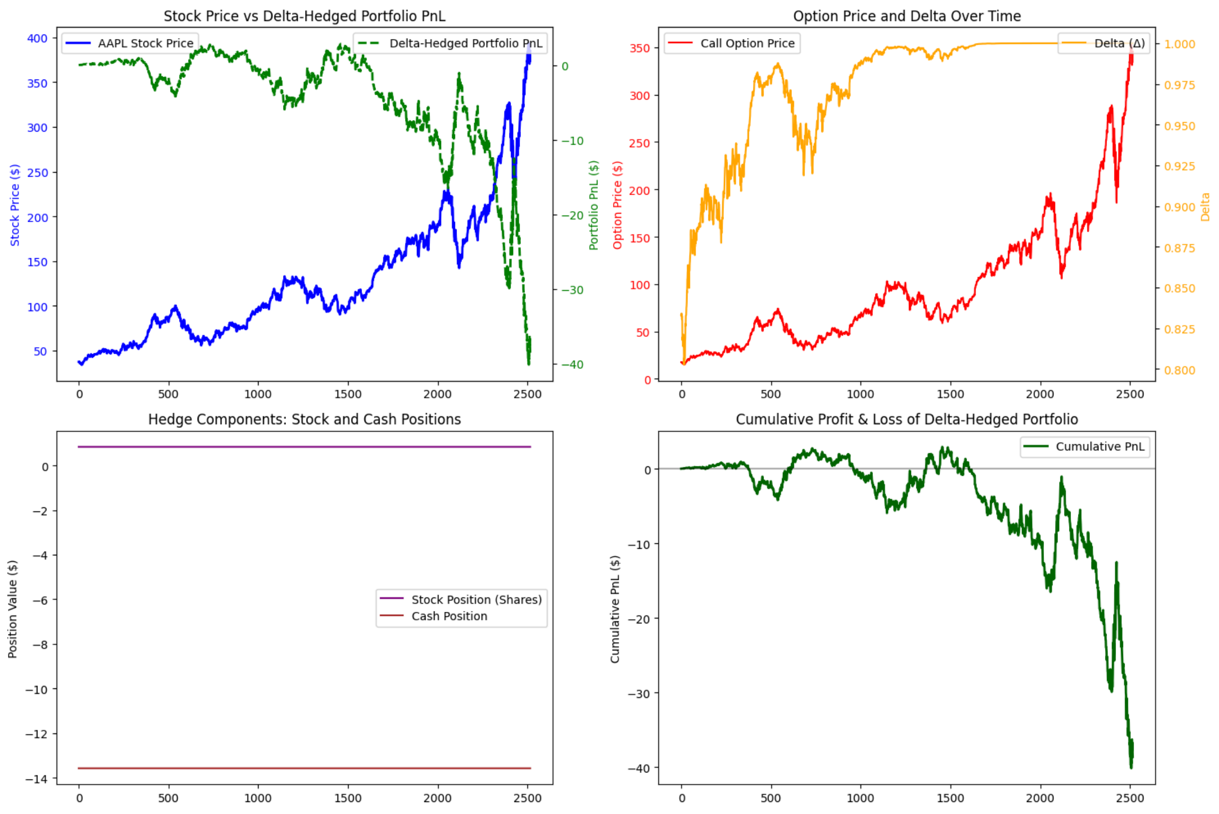
```
Apple data shape: (2517, 7)
Date range: 0 to 2516
```

```
In [6]: #This method computes the Black-Scholes price and delta
        class BlackScholes:
            @staticmethod
            def calculate(S, K, T, r, sigma, option_type="call"):
                """
                S: current stock price
                K: strike price
                T: time to expiration (in years)
                r: risk-free rate
                sigma: volatility
                """
                if T <= 0:
                    if option_type == "call":
                        return max(S - K, 0)
                    else:
                        return max(K - S, 0)

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

                if option_type == "call":
                    price = S * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2)
                    delta = norm.cdf(d1)
                else:  # put
                    price = K * np.exp(-r * T) * norm.cdf(-d2) - S * norm.cdf(-d1)
                    delta = norm.cdf(d1) - 1

                return price, delta

        # Example parameters
        S0 = aapl_data['Close'].iloc[0]  # Initial stock price
        K = S0 * 1.05  # 5% out-of-the-money strike
        r = 0.05  # 5% risk-free rate
        sigma = 0.25  # 25% volatility
        T = 30/365  # 30 days to expiration
```

```
In [7]: def delta_hedge_simulation(stock_prices, initial_params):
            """
            Simulate delta-hedging strategy
```

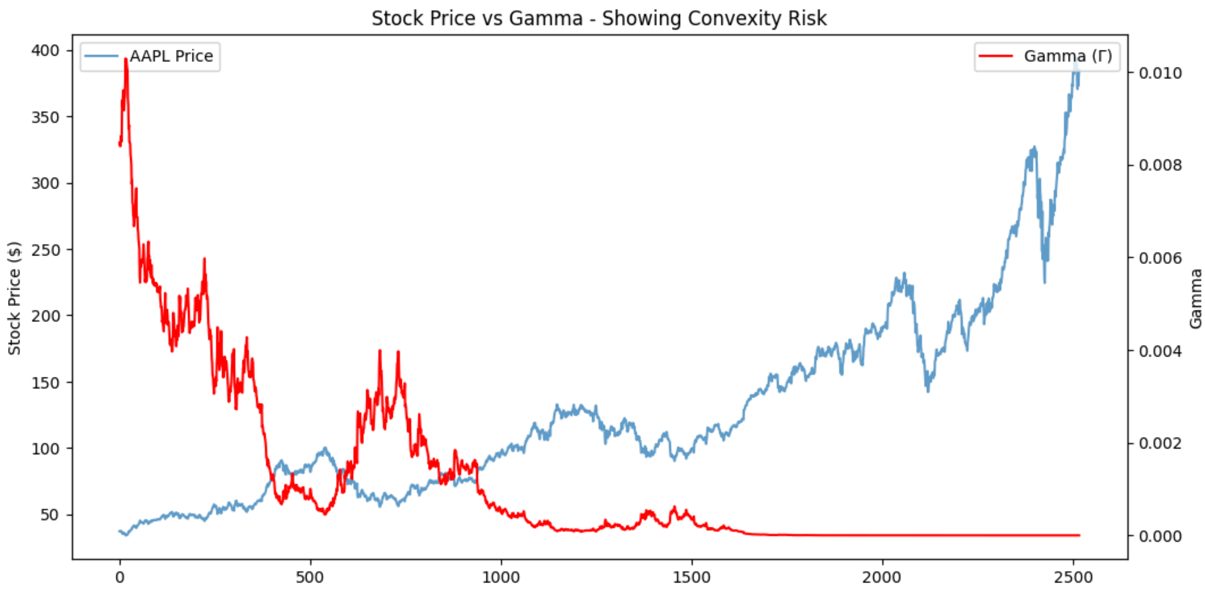
```
    """
    K = initial_params['strike']
    r = initial_params['risk_free_rate']
    sigma = initial_params['volatility']
    total_days = len(stock_prices)

    # Storage arrays
    portfolio_values = []
    option_prices = []
    deltas = []
    stock_positions = []
    cash_positions = []
    pnl = []

    # Initial setup
    initial_price = stock_prices[0]
    T_initial = total_days / 252  # Convert to years

    # Calculate initial option price and delta
    option_price, delta = BlackScholes.calculate(
        initial_price, K, T_initial, r, sigma, "call"
    )

    # Initial portfolio: Short 1 call option + Long delta shares
    stock_position = delta
    cash = option_price - (delta * initial_price)  # Premium received minus st

    portfolio_values.append(0)  # Start at zero
    option_prices.append(option_price)
    deltas.append(delta)
    stock_positions.append(stock_position)
    cash_positions.append(cash)
    pnl.append(0)

    # Daily hedging simulation
    for i in range(1, len(stock_prices)):
        current_price = stock_prices[i]
        days_remaining = total_days - i
        T_current = days_remaining / 252

        # Calculate current option price and delta
        if days_remaining > 0:
            option_price, delta = BlackScholes.calculate(
                current_price, K, T_current, r, sigma, "call"
            )
        else:
            # At expiration
            option_price = max(current_price - K, 0)
            delta = 1 if current_price > K else 0

        # STATIC HEDGE: We don't rebalance (key difference from dynamic hedgin
        # Portfolio value = Stock position + Cash - Option liability
        stock_value = stock_positions[0] * current_price  # Using initial posi
        portfolio_value = stock_value + cash_positions[0] - option_price

        portfolio_values.append(portfolio_value)
        option_prices.append(option_price)
        deltas.append(delta)
        stock_positions.append(stock_positions[0])  # No change in static hedg
        cash_positions.append(cash_positions[0])    # No change in static hedg
        pnl.append(portfolio_value)

    return {
        'portfolio_values': portfolio_values,
        'option_prices': option_prices,
        'deltas': deltas,
        'stock_positions': stock_positions,
        'cash_positions': cash_positions,
        'pnl': pnl
    }
```

In [8]:
```
def run_complete_analysis():
    # Prepare data
    stock_prices = aapl_data['Close'].values

    # Simulation parameters
    params = {
        'strike': stock_prices[0] * 1.05,  # 5% OTM
        'risk_free_rate': 0.05,
        'volatility': 0.25
    }

    # Run simulation
    results = delta_hedge_simulation(stock_prices, params)

    # Create subplots
    fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(15, 10))

    # Plot 1: Stock Price vs Portfolio PnL
    ax1.plot(stock_prices, label='AAPL Stock Price', color='blue', linewidth=2
    ax1.set_ylabel('Stock Price ($)', color='blue')
    ax1.tick_params(axis='y', labelcolor='blue')
    ax1.legend(loc='upper left')

    ax1_twin = ax1.twinx()
    ax1_twin.plot(results['pnl'], label='Delta-Hedged Portfolio PnL',
                  color='green', linewidth=2, linestyle='--')
    ax1_twin.set_ylabel('Portfolio PnL ($)', color='green')
    ax1_twin.tick_params(axis='y', labelcolor='green')
    ax1_twin.legend(loc='upper right')
    ax1.set_title('Stock Price vs Delta-Hedged Portfolio PnL')

    # Plot 2: Option Price and Delta
    ax2.plot(results['option_prices'], label='Call Option Price', color='red')
    ax2.set_ylabel('Option Price ($)', color='red')
    ax2.tick_params(axis='y', labelcolor='red')
    ax2.legend(loc='upper left')

    ax2_twin = ax2.twinx()
    ax2_twin.plot(results['deltas'], label='Delta (Δ)', color='orange')
    ax2_twin.set_ylabel('Delta', color='orange')
    ax2_twin.tick_params(axis='y', labelcolor='orange')
    ax2_twin.legend(loc='upper right')
    ax2.set_title('Option Price and Delta Over Time')

    # Plot 3: Hedge Components
    ax3.plot(results['stock_positions'], label='Stock Position (Shares)', colo
    ax3.plot(results['cash_positions'], label='Cash Position', color='brown')
    ax3.set_ylabel('Position Value ($)')
    ax3.legend()
    ax3.set_title('Hedge Components: Stock and Cash Positions')

    # Plot 4: Cumulative PnL Breakdown
    daily_returns = np.diff(results['pnl'])
    cumulative_pnl = np.cumsum(daily_returns)
    ax4.plot(cumulative_pnl, label='Cumulative PnL', color='darkgreen', linewi
    ax4.axhline(y=0, color='black', linestyle='-', alpha=0.3)
    ax4.set_ylabel('Cumulative PnL ($)')
    ax4.legend()
    ax4.set_title('Cumulative Profit & Loss of Delta-Hedged Portfolio')

    plt.tight_layout()
    plt.show()

    return results

# Run the analysis
results = run_complete_analysis()
```



In [9]:
```
def calculate_gamma(S, K, T, r, sigma):
    """Calculate Gamma for the option"""
    if T <= 0:
        return 0

    d1 = (np.log(S / K) + (r + 0.5 * sigma**2) * T) / (sigma * np.sqrt(T))
    gamma = norm.pdf(d1) / (S * sigma * np.sqrt(T))
    return gamma

def gamma_analysis(stock_prices, params):
    """Analyze gamma risk in the hedge"""
    gammas = []
    for i, price in enumerate(stock_prices):
        days_remaining = len(stock_prices) - i
        T_current = days_remaining / 252
        gamma = calculate_gamma(price, params['strike'], T_current,
                                params['risk_free_rate'], params['volatility'])
        gammas.append(gamma)

    return gammas

# Add gamma analysis
gammas = gamma_analysis(aapl_data['Close'].values, {
    'strike': aapl_data['Close'].iloc[0] * 1.05,
    'risk_free_rate': 0.05,
    'volatility': 0.25
})
```

```
plt.figure(figsize=(12, 6))
plt.plot(aapl_data['Close'].values, label='AAPL Price', alpha=0.7)
plt.ylabel('Stock Price ($)')
plt.legend(loc='upper left')

plt.twinx()
plt.plot(gammas, label='Gamma (Γ)', color='red')
plt.ylabel('Gamma')
plt.legend(loc='upper right')
plt.title('Stock Price vs Gamma - Showing Convexity Risk')
plt.show()
```



In [ ]:
```
def calculate_performance_metrics(results, stock_prices):
    """Calculate key performance metrics"""
    pnl = results['pnl']
    stock_returns = np.diff(stock_prices) / stock_prices[:-1]
    portfolio_returns = np.diff(pnl) / (np.mean(np.abs(pnl)) + 1e-10)  # Avoic

    metrics = {
        'Total PnL': pnl[-1] - pnl[0],
        'PnL Volatility': np.std(np.diff(pnl)),
        'Max PnL': np.max(pnl),
        'Min PnL': np.min(pnl),
        'Final Hedge Error': pnl[-1],  # Should be close to zero for perfect h
        'Stock Volatility': np.std(stock_returns),
        'Portfolio Volatility': np.std(portfolio_returns)
    }

    print("=== DELTA-HEDGING PERFORMANCE METRICS ===")
    for key, value in metrics.items():
        print(f"{key}: {value:.4f}")

    return metrics

# Calculate metrics
```

```python
metrics = calculate_performance_metrics(results, aapl_data['Close'].values)
```

```
=== DELTA-HEDGING PERFORMANCE METRICS ===
Total PnL: -38.6307
PnL Volatility: 0.4723
Max PnL: 2.9272
Min PnL: -40.1424
Final Hedge Error: -38.6307
Stock Volatility: 0.0173
Portfolio Volatility: 0.0918
```