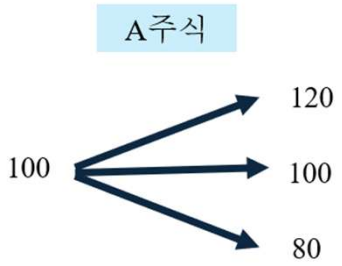
Pricing and Hedging of Derivatives

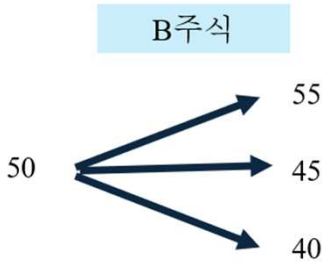
https://github.com/HST0077/HYOTC

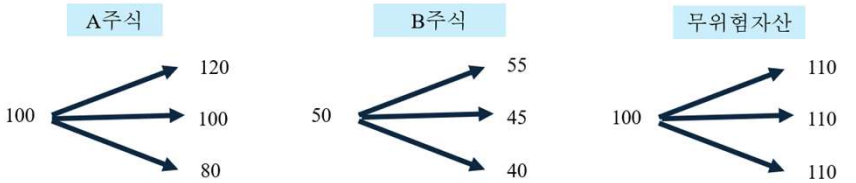
Review

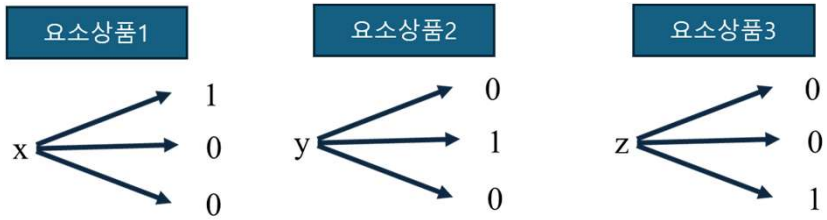
Option pricing using state prices

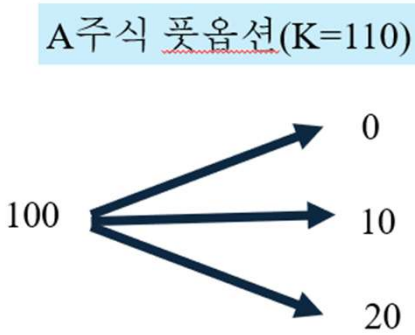
Q. 현재 무위험 이자율은 10%, A주식의 현재가는 100, B주식의 현재가는 50이라고 한다. 1년 후에 A주식과 B주식의 가격은 오른쪽과 같은 경우만 존재한다. 행사가 110인 A주식의 풋옵션의 현재가는 얼마여야 하는가?











A 풋옵션의 가격:

10y+20z=4.5454

Monte Carlo Simulation for Put Option

$$dN_t = rN_t dt + \sigma N_t dW_t$$

$$N_t = N_0 e^{\left(r-\frac{1}{2}\sigma^2\right)t+\sigma W_t}$$

```python
# 풋옵션 함수 작성

def MC_Put(S,K,T,r,sigma,n_sim=100000):
    import numpy as np
    # 난수 생성 (표준 정규분포)
    np.random.seed(11)
    Z = np.random.randn(n_sim)
    # 만기가격 도출
    ST = S * np.exp((r - 0.5 * sigma**2) * T\
                    + sigma * np.sqrt(T) * Z)
    # 풋옵션의 페이오프 계산
    payoff = np.maximum(K-ST, 0)
    # 현재가치 할인 적용 (할인율: e^(-rT))
    put_price = np.exp(-r * T) * np.mean(payoff)
    return put_price
```

```python
MC_Greeks(MC_Put, 100, 100, 1, 0.05, 0.3)
```

```
{'Price': 9.275129383003724,
 'Delta (Δ)': -0.37516958155066504,
 'Gamma (Γ)': 0.01162181001753816,
 'Theta (Θ)': -0.009135373917150295,
 'Vega (V)': 0.3772682484415787,
 'Rho (ρ)': -0.004678548225085848}
```

```python
def MC_Greeks(fun, S, K, T, r, sigma):
    # Delta (Δ)
    eps=S*sigma*0.01
    C_plus = fun(S + eps, K, T, r, sigma)
    C_minus = fun(S - eps, K, T, r, sigma)
    Delta = (C_plus - C_minus) / (2 * eps)
    # Gamma (Γ)
    C_0 = fun(S, K, T, r, sigma)
    Gamma = (C_plus - 2 * C_0 + C_minus) / (eps ** 2)
    # Vega (V)
    eps=sigma*0.01
    C_sigma_plus = fun(S, K, T, r, sigma + eps)
    C_sigma_minus = fun(S, K, T, r, sigma - eps)
    Vega = (C_sigma_plus - C_sigma_minus) / (2 * eps)
    Vega = 0.01*Vega # 1% 베가로 변환
    # 1day Theta (Θ)
    eps=0.01
    C_t = fun(S, K, T - eps, r, sigma)
    Theta = (1/365)*(C_t - C_0) / eps
    # 1bp Rho (ρ)
    C_r_plus = fun(S, K, T, r + eps, sigma)
    C_r_minus = fun(S, K, T, r - eps, sigma)
    Rho = 0.0001* (C_r_plus - C_r_minus) / (2 * eps)
    return {
        "Price": C_0,
        "Delta (Δ)": Delta,
        "Gamma (Γ)": Gamma,
        "Theta (Θ)": Theta,
        "Vega (V)": Vega,
        "Rho (ρ)": Rho,
    }
```

1 Stock Step Down

6개월 마다 휴일이 아닌 영업일 날짜 연산

```python
# 대한민국 공휴일 불러오기
kor_holidays = list(holidays.KOR(years=range(2025, 2031)).keys())
kor_holidays = pd.to_datetime(kor_holidays)

# 오늘 날짜
today = pd.to_datetime(datetime.today().date())
six_months_later = today + pd.DateOffset(months=6)

# 6개월마다의 기준일 생성
month_starts = pd.date_range(start=six_months_later, periods=6, freq=pd.DateOffset(months=6))

# 각 기준일에서 해당 월의 첫 번째 영업일(공휴일 제외) 추출
business_days = []
for dt in month_starts:
    # 시작 날짜부터 한 달간의 영업일 추출
    month_range = pd.date_range(start=dt, end=dt + pd.offsets.MonthEnd(0), freq='B')
    # 영업일이 대한민국의 휴일이 아닌 것을 다시 추출출
    valid_days = [d for d in month_range if d not in kor_holidays]
    if valid_days: # 자료가 있다면
        business_days.append(valid_days[0])

# 결과 출력
print(business_days)
```

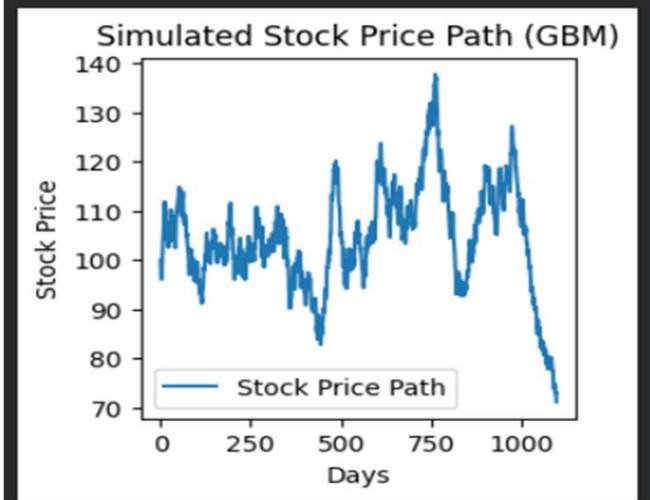
```python
from datetime import datetime

# 날짜 차이 계산 (단위: 일수)
days_diff = [(d - today).days for d in business_days]

# DataFrame 생성
df = pd.DataFrame({
    '영업일': [d.strftime('%Y-%m-%d') for d in business_days],
    '오늘까지 남은 일수': days_diff})
df
```

	영업일	오늘까지 남은 일수
0	2026-05-18	181
1	2026-11-18	365
2	2027-05-18	546
3	2027-11-18	730
4	2028-05-18	912
5	2028-11-20	1098

주가 path 생성하기

```python
# 주가 경로 생성
N=df['오늘까지 남은 일수'].iloc[-1] # 만기까지의 날짜수
sim=1000 # 시뮬레이션 회수
r,sigma=0.05, 0.3
dt=1/365

np.random.seed(111)
W = np.random.randn(N,sim)  # 표준 정규분포 난수

# log(S) 행렬 만들기
lnS=(r - 0.5 * sigma**2) * dt + sigma * np.sqrt(dt) * W

# 앞에 붙이기
lnS = np.insert(lnS,0,np.log(100), axis=0)
# 누적합 구하기
S=np.exp(np.cumsum(lnS,axis=0))
```

```python
# 첫번째 시뮬레이션 결과 시각화
plt.figure(figsize=(3,3))
plt.plot(S[:,0], label="Stock Price Path")
plt.xlabel("Days")
plt.ylabel("Stock Price")
plt.title("Simulated Stock Price Path (GBM)")
plt.legend()
plt.show()
```



Step Down 평가 함수 만들기: 입력변수 설정

```
# 스텝다운 상품 평가 정보
S0=100
kijun=100
r=0.035
q=0
sigma, barrier, dummy, sim=0.3, 0.65, 0.132,10000
K=[0.95,0.95,0.95,0.90,0.90,0.85]
T=np.array(df['오늘까지 남은 일수'])
c=np.array([1,2,3,4,5,6])*0.022

SD_Istar_GPU(S0, kijun, K, T, c, r, q, sigma, barrier, dummy, sim)
```

Step Down 평가 함수 만들기: 주가 path생성

```python
def SD_Istar_GPU(S0, kijun, K, T, c, r, q, sigma, barrier, dummy, sim):
    import cupy as cp
    from scipy.stats import norm  # 정규분포 적합은 여전히 CPU에서 수행
    import numpy as np

    """
    ELS (Equity-Linked Securities) 조기상환 및 만기상환 시뮬레이션 (GPU 버전)
    """

    N = T[-1]  # 전체 만기까지의 날짜 수
    dt = 1 / 365  # 하루 단위
    cp.random.seed(111)
    W = cp.random.randn(N, sim)  # 표준 정규분포 난수 생성

    # 로그 수익률 경로 생성
    lnS = (r - 0.5 * sigma**2) * dt + sigma * cp.sqrt(dt) * W

    # 시작값 log(S0)를 앞에 추가
    lnS = cp.concatenate([cp.full((1, sim), cp.log(S0)), lnS], axis=0)

    # 누적합 후 지수화하여 주가 경로 생성
    S_path = cp.exp(cp.cumsum(lnS, axis=0))
```

Step Down 평가 함수 만들기: 조기상환 판단

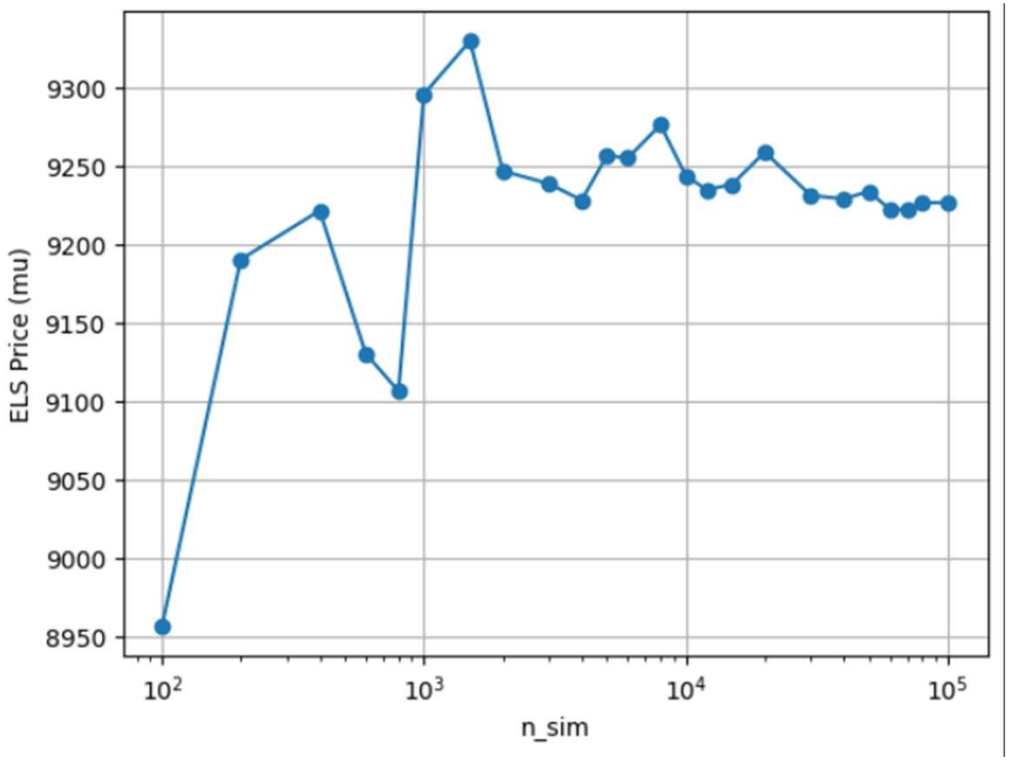
```
# 수익률 행렬 계산
R = S_path / kijun

# 초기 옵션 가격 배열
Price = cp.zeros(sim)

EN = len(K)

# 조기상환여부 판단
for i in range(EN):
    # 현재 가격이 0이고, 조기상환일(T[i])의 수익율이 행사가 이상인 인덱스 추출
    cond = (Price == 0) & (R[T[i], :] >= K[i])
    # 해당 인덱스의 값을, 조기상환쿠폰(c[i])으로 상환되며 이를 현재가로 변환
    Price = cp.where(cond, 10000 * (1 + c[i]) * cp.exp(-r * T[i] / 365), Price)
```

Step Down 평가 함수 만들기: 만기손실 판단

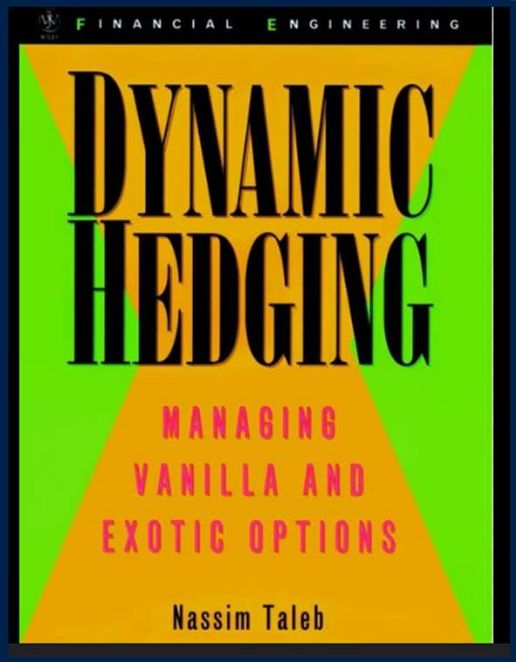
```
# 아직 상환되지 않은 시뮬레이션 인덱스 찾기
check = (Price == 0)

# 배리어 하회 여부 확인
min_R = cp.min(R, axis=0) # 각 시뮬레이션에서 최소 수익율 인덱스 추출
barrier_hit = (min_R < barrier) & check # 미상환된 종목주 최소 수익율일 배리어보다 작은 인덱스 추출
no_barrier_hit = ~barrier_hit & check # 미상환된 종목주 최소 수익율일 배리어를 터치하지 않은 인덱스 추출

# 배리어 하회한 경우
Price = cp.where(barrier_hit, 10000 * R[-1, :] * cp.exp(-r * T[-1] / 365), Price)

# 배리어를 한 번도 안 맞은 경우
Price = cp.where(no_barrier_hit, 10000 * (1 + dummy) * cp.exp(-r * T[-1] / 365), Price)

# GPU에서 CPU로 데이터 이동하여 정규분포 적합
Price_cpu = cp.asnumpy(Price)
mu, s = norm.fit(Price_cpu)

return mu
```

Step Down 평가 함수 만들기: 수렴도 살펴보기

```
Sims=[100,200,400,600,800,1000,1500,2000,3000,4000,
      5000,6000,8000,10000,12000,15000,20000,30000,
      40000,50000,60000,70000,80000,100000]

Price_Sims=[]
for sim in Sims:
    res=SD_Istar_GPU(S0, kijun, K, T, c, r, q, sigma, barrier, dummy, sim)
    Price_Sims.append(res)

plt.figure()
plt.plot(Sims, Price_Sims, marker='o')
plt.xscale('log')  # 로그스케일로 보면 수렴이 잘 보임
plt.xlabel("n_sim")
plt.ylabel("ELS Price (mu)")
plt.grid(True)
plt.show()
```



Hedging of Derivatives



풋옵션의 이론가 평가

```python
from hst_funcs.finanace.Equity import BS
import numpy as np
from datetime import datetime

# 파라미터 설정
S0 = 586.73
K = 535
today= datetime(2025,11,12)
maturity=datetime(2025,12,11)

# 날짜 차이 계산 (단위: 일수)
days_diff = (maturity - today).days

T = days_diff/365
r = 0.0265
sigma = 0.3

opt0 = BS.bs_put(S0, K, T, r, sigma)
print('옵션 현재가치: ', opt0)
```

```
옵션 현재가치:  3.1419035008890432
```

2025년 11월12일, 301WC535 종목의 종가는 5.88

이론가보다 현재 종가 가격이 비싸기 때문에 해당 종목을 매도하는게 유리할 수 있음

그러나, 수익을 확정할 수 있는 것은 아님!!!

WHY?

내재변동성(implied volatility)

```python
# 이분법에 의한 내재변동성계산
def implied_vol_bisection(C_mkt, S, K, r, q, T,
                          sigma_low=1e-6, sigma_high=5.0, tol=1e-8, max_iter=100):
    """
    C_mkt: 시장에서 관측된 옵션 프라이스
    sigma_low / sigma_high: 변동성 탐색 구간
    """
    for i in range(max_iter):
        sigma_mid = 0.5 * (sigma_low + sigma_high)
        price = BS.bs_put(S0, K, T, r, sigma_mid)

        if abs(price - C_mkt) < tol:
            return sigma_mid

        # 가격이 시장가격보다 작으면 sigma를 올려야 함
        if price < C_mkt:
            sigma_low = sigma_mid
        else:
            sigma_high = sigma_mid

    # 수렴 실패 시 중간값 반환
    return sigma_mid
```

```python
C_mkt,q=5.88, 0
imvol=implied_vol_bisection(C_mkt, S0, K, r, q, T)
print(f'내재변동성: {imvol*100:.2f}%')
```

```
내재변동성: 37.19%
```

```python
opt0 = BS.bs_put(S0, K, T, r, imvol)
print('옵션 현재가치: ', opt0)
```

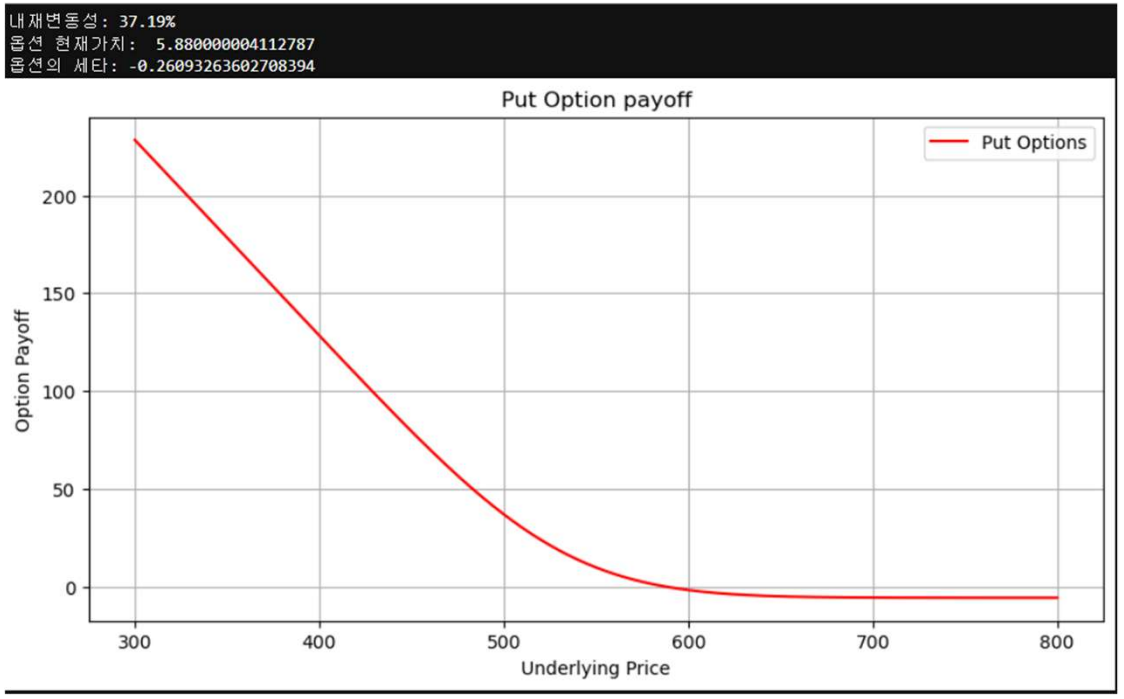
```
옵션 현재가치:  5.880000004112787
```

풋옵션의 이론가 그래프

```
# 파라미터 설정
S0 = 586.73
S_max = 800
K = 535
today= da
maturity=

# 날짜 차
days_diff

T = days_
r = 0.026

C_mkt,q=5
imvol=imp
print(f'L
```

```
sigma = imvol
N = 500 # 기초자산 격자

opt_values = np.zeros(N+1)
opt_payoff = np.zeros(N+1)

opt0 = BS.bs_put(S0, K, T, r, sigma)
print('옵션 현재가치: ', opt0)
theta = BS.put_theta(S0, K, T, r, sigma)
print('옵션의 세타:',theta)

S = np.linspace(300, S_max, N+1)

for i,s in enumerate(S):
    opt_values[i]=BS.bs_put(s, K, T, r, sigma)

# 하루 뒤의 option payoff
opt_payoff=opt_values-opt0-theta
```

```
내재변동성: 37.19%
옵션 현재가치:  5.880000004112787
옵션의 세타: -0.26093263602708394
```



Volatility Smile

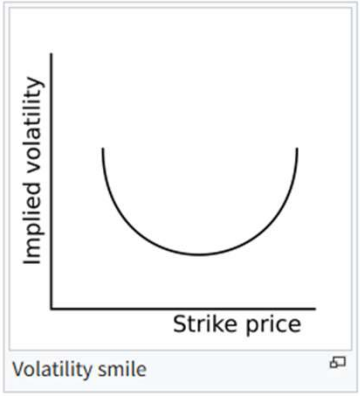
Volatility smile

4 languages

Article Talk | Read Edit View history Tools

From Wikipedia, the free encyclopedia

Volatility smiles are implied volatility patterns that arise in pricing financial options. It is a parameter (implied volatility) that needs to be modified for the Black–Scholes formula to fit market prices. Generally, for a given expiration, options whose strike price differs substantially from the underlying asset's forward price tend to have prices that deviate from their expected prices using a constant-volatility model based on the at-the-money (strike price near the underlying's forward price).



Volatility smile

Graphing implied volatilities against strike prices for a given expiry produces a skewed "smile" instead of the expected flat surface. The pattern differs across various markets. Equity options traded in American markets did not show a significant volatility smile before the Crash of 1987 but began showing one afterwards.[1] It is believed that investor reassessments of the probabilities of fat-tail have led to higher prices for out-of-the-money options. This anomaly implies deficiencies in the standard Black–Scholes option pricing model which assumes constant volatility and log-normal distributions of underlying asset returns. Empirical asset returns distributions, however, tend to exhibit fat-tails (kurtosis) and skew. Modelling the volatility smile is an active area of research in quantitative finance, and better pricing models such as the stochastic volatility model partially address this issue.

특정만기에 대해 ATM을 기준으로 외가격 call option의 내재변동성과 외가격 put option의 내재변동성을 연결한 곡선을 의미

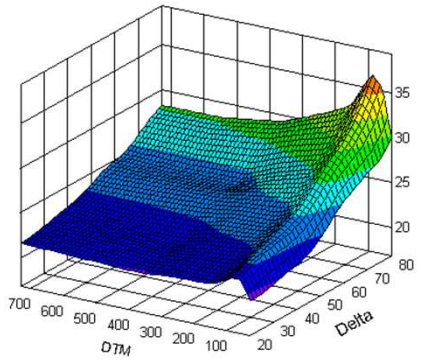
Volatility Surface

Implied volatility surface [edit]

It is often useful to plot implied volatility as a function of both strike price and time to maturity.[4] The result is a two-dimensional curved surface plotted in three dimensions whereby the current market implied volatility (*z*-axis) for all options on the underlying is plotted against the strike price (*y*-axis) and time to maturity (*x*-axis "DTM"). This defines the **absolute implied volatility surface**; changing coordinates so that the strike price is replaced by delta yields the **relative implied volatility surface**.

The implied volatility surface simultaneously shows both volatility smile and term structure of volatility. Option traders use an implied volatility plot to quickly determine the shape of the implied volatility surface, and to identify any areas where the slope of the plot (and therefore relative implied volatilities) seems out of line.

The graph shows an implied volatility surface for all the put options on a particular underlying stock price. The *z*-axis represents implied volatility in percent, and *x* and *y* axes represent the option delta, and the days to maturity. Note that to maintain put–call parity,[note 1] puts must have the same implied volatility as must have the same implied volatility as calls of the same strike and expiration date. For this surface, we can see that the underlying symbol has both volatility skew (a tilt along the delta axis), as well as a volatility term structure, indicating an anticipated event in the near future.



만기변화에 따른 volatility smile 커브를 연결하면 하나의 곡면으로 나타낼 수 있음.

중립 델타 수량

```
# 중립델타수량
# 코스피200 옵션 1pt의 가치 250,000
# 코스피200 선물 1pt의 가치 250,000
# 11월12일 선물 종가: 587.70

delta = BS.put_delta(S0, K, T, r, imvol)
print('매수 1계약 옵션의 델타: ', delta)
print('매수 100계약 옵션의 델타: ', delta*100)
```

```
매수 1계약 옵션의 델타:  -0.1702841925792607
매수 100계약 옵션의 델타:  -17.02841925792607
```

매수 풋옵션 100계약을 헤지하기 위해서는 선물 17.03계약을 매수하여야 부호가 상쇄된다.

옵션의 이론가와 실제 가격의 괴리

```
# 11월 13일 KOSPI200 현물 종가: 588.65
# 11월 13일 KOSPI200 선물 종가: 590.75
# 11월 13일 301WC535 풋옵션 종가: 4.36

# 파라미터 설정
S0 = 588.65
K = 535
today= datetime(2025,11,13)
maturity=datetime(2025,12,11)

# 날짜 차이 계산 (단위: 일수)
days_diff = (maturity - today).days

T = days_diff/365
r = 0.0265

opt0 = BS.bs_put(S0, K, T, r, imvol)
print('옵션 11월13일 종가 기준 현재 이론 가치: ', opt0)
C_mkt,q=4.36, 0
imvol=implied_vol_bisection(C_mkt, S0, K, r, q, T)
print(f'다음날 종가에 형성된 내재변동성: {imvol*100:.2f}%')
```

```
옵션 11월13일 종가 기준 현재 이론 가치:  5.305973116774908
내재변동성: 34.71%
```

11월13일은 11월12일 종가 대비 현물 가격이 상승하면서 풋옵션의 가치와 함께 내재변동성이 동시에 하락한 것을 관찰할 수 있음

델타 중립 헤지 효과

```
theta = BS.put_theta(S0, K, T, r, imvol)
print('옵션 100계약 매수시 하루 후 손실예상 금액: {:,.0f}원'.format(theta*100*250000))
```

```
내재변동성: 37.19%
옵션 100계약 매수시 하루 후 손실예상 금액: -6,523,316원
```

```
# 이론적 델타 중립 효과
print('옵션 100계약 매수 손실금액:{:,.0f}원'.format(250000*100*(5.88-5.30597)))
print('델타 중립 현물 계약 매수 이득금액:{:,.0f}원'.format(250000*17.03*(588.65-586.73)))
print('델타 중립 손익:{:,.0f}원'.format(-250000*100*(5.88-5.30597)+250000*17.03*(588.65-586.73)))
```

```
옵션 100계약 매수 손실금액:14,350,750원
델타 중립 현물 계약 매수 이득금액:8,174,400원
델타 중립 손익:-6,176,350원
```

```
# 실제 델타 중립 효과
print('옵션 100계약 매수 손실금액:{:,.0f}원'.format(250000*100*(5.88-4.36)))
print('델타 중립 선물 계약 매수 이득금액:{:,.0f}원'.format(250000*17.03*(590.75-587.70)))
print('델타 중립 손익:{:,.0f}원'.format(-250000*100*(5.88-4.36)+250000*17.03*(590.75-587.70)))
```

```
옵션 100계약 매수 손실금액:38,000,000원
델타 중립 선물 계약 매수 이득금액:12,985,375원
델타 중립 손익:-25,014,625원
```

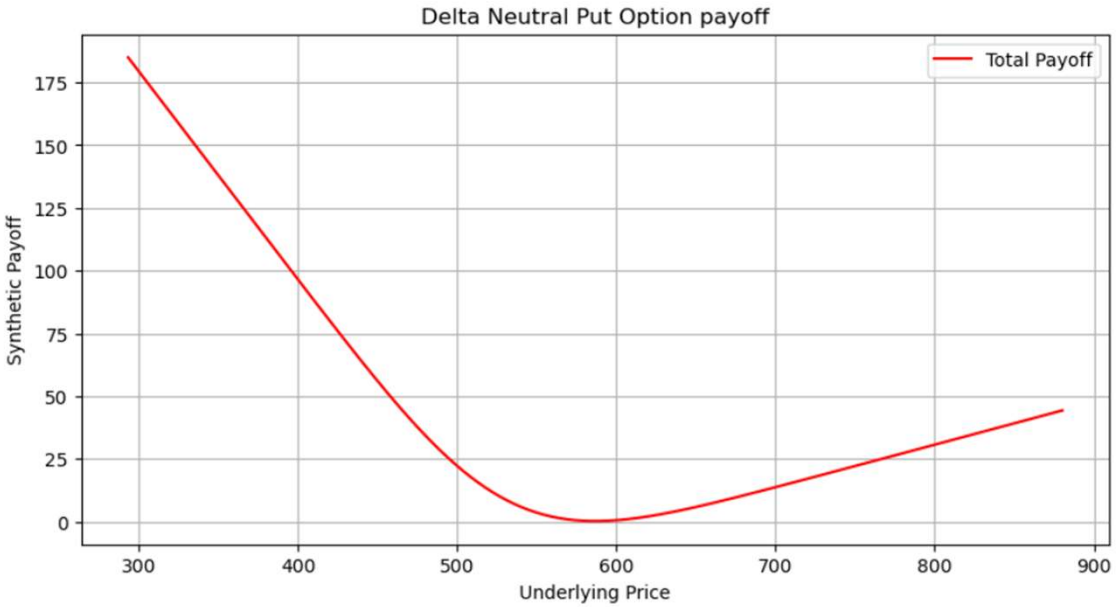
이론적으로는 세타 손실의 일부는 커버했음. 실질적으로 옵션 내재변동성 변화, 선물가격 괴리 등에 의해 손실 확대

옵션매수 델타 중립 이론 Payoff

```
plimit=0.5

# 가격의 최소값과 최대값을 계산
min_price = S0 * (1 - plimit)
max_price = S0 * (1 + plimit)

# min_pr
S = np.l

opt_valu
opt_payo

opt0 = B
print('옵
theta =
print('옵
```

```
# with delta hedgaing
delta_payoff = np.zeros(N+1)
delta = BS.put_delta(S0, K, T, r, imvol)
delta_payoff = (S - S0) * (-delta)


for i,s in enumerate(S):
    opt_values[i]=BS.bs_put(s, K, T, r, imvol)

# 하루 뒤의 option payoff
opt_payoff=opt_values-opt0-theta
```

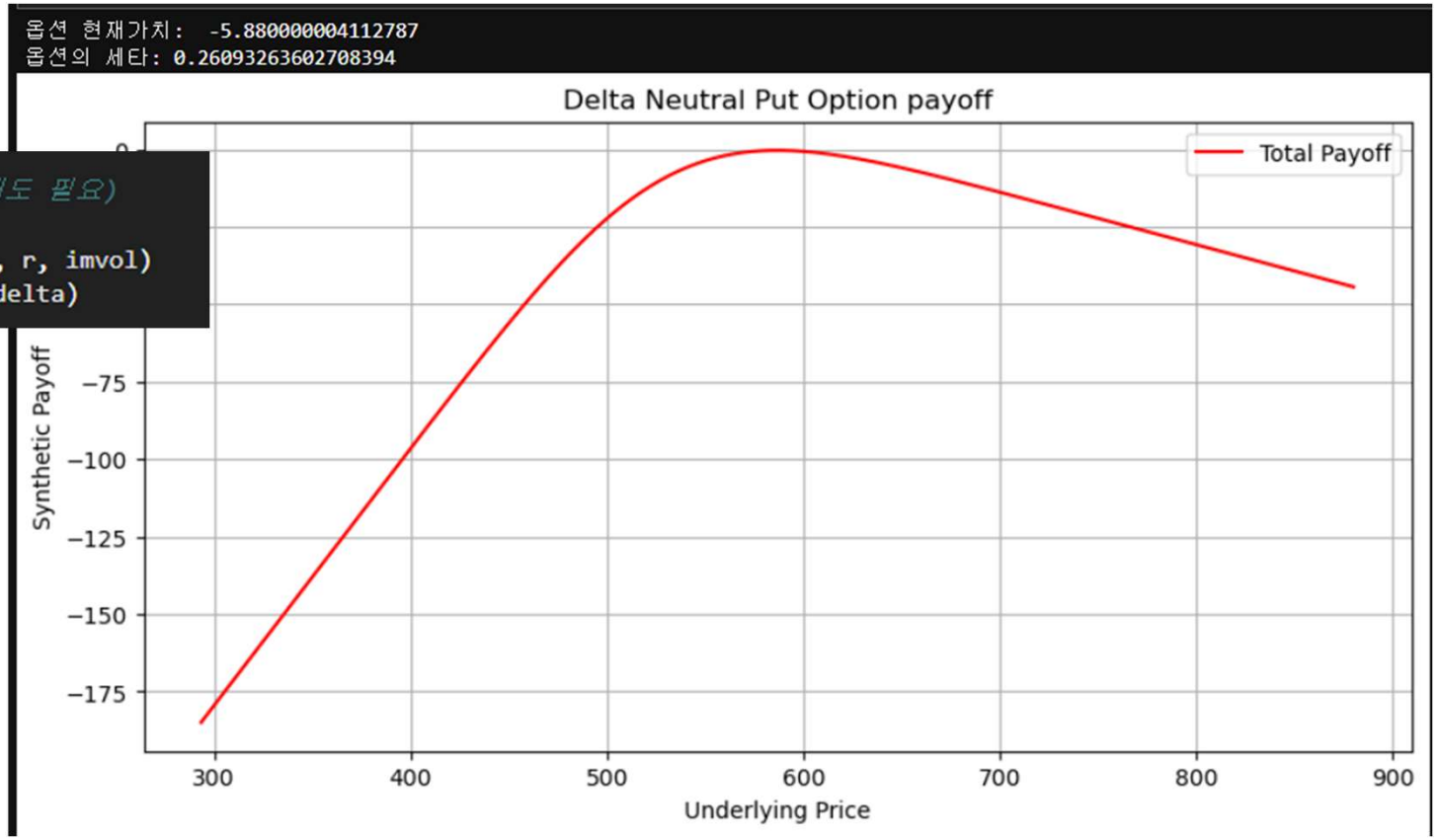
```
# with delta hedgaing
total_payoff = np.zeros(N+1)
total_payoff= delta_payoff+opt_payoff
```



옵션매도 델타 중립 이론 Payoff

```
# with delta hedgaing (선물 매도 필요)
delta_payoff = np.zeros(N+1)
delta = BS.put_delta(S0, K, T, r, imvol)
delta_payoff = -(S - S0) * (-delta)
```

```
옵션 현재가치:  -5.880000004112787
옵션의 세타: 0.26093263602708394
```



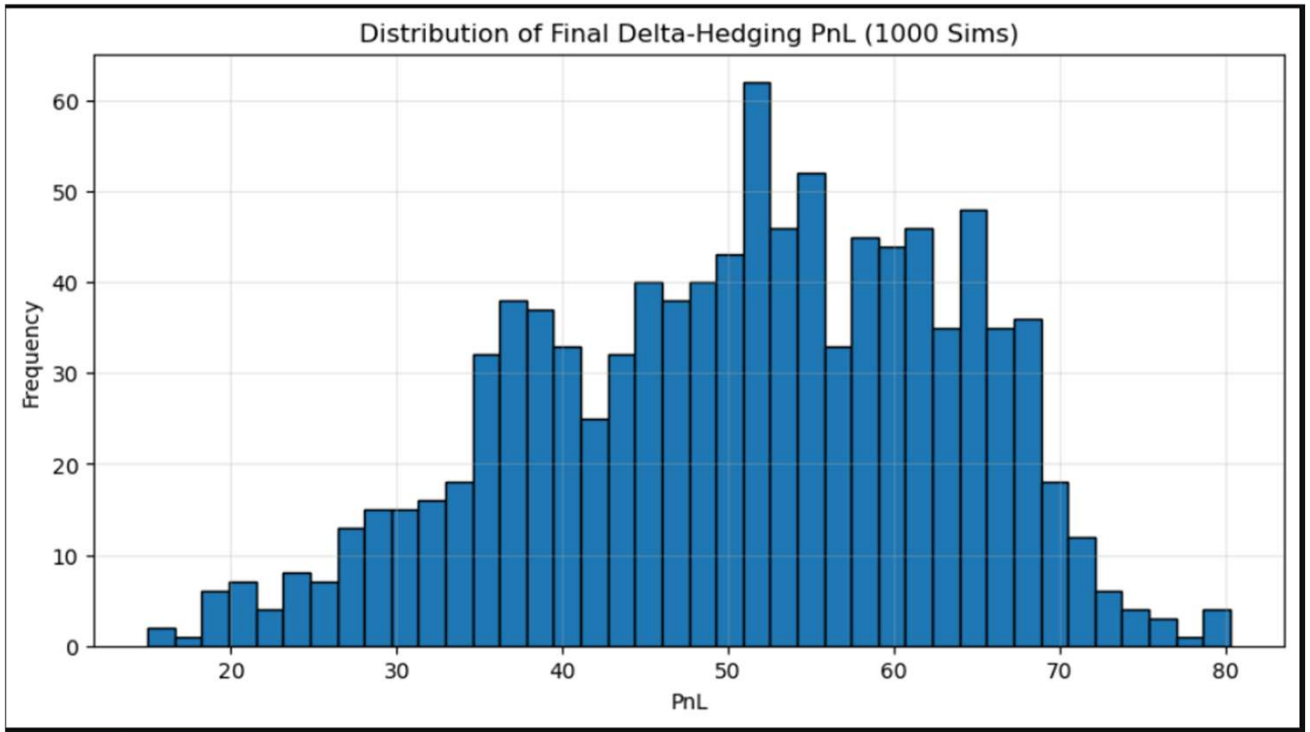
델타헤징 포트폴리오 시뮬레이션

```python
# 첫 번째 path만 사용
S_path = S[:, 0]                    # shape: (N+1,)

# 날짜
dates =

n_steps
opt_val
deltas
hedge_p

# t=0
T0 = (m
opt_val

# 초기
hedge_p

# PnL
opt_pnl
hedge_p
total_p
```

```python
for t in range(1, n_steps):
    # 잔존만기 (년)
    T_remain = (maturity - dates[t]).days / 365
    T_remain = max(T_remain, 1)

    # 현재 옵션 가치와 델타
    opt_values[t], deltas[t] = BS.bs_put(S_path[t], K, T_remain, r, sigma

    # (t-1) -> t 구간 옵션 PnL
    opt_pnl[t] = opt_values[t] - opt_values[t-1]

    # (t-1) 시점에 보유하던 헤지 포지션으로 기초자산 PnL
    hedge_pnl[t] = hedge_pos[t-1] * (S_path[t] - S_path[t-1])

    # 전체 PnL
    total_pnl[t] = opt_pnl[t] + hedge_pnl[t]

    # t 시점에 새 델타에 맞게 리밸런싱
    hedge_pos[t] = -deltas[t]

# 누적 PnL
cum_total_pnl = np.cumsum(total_pnl)
```

Date	Spot	OptionValue	Delta	HedgePosition	OptionPnL	HedgePnL	TotalPnL	CumTotalPnL
2025-11-12	586.730000	5.880000	-0.170284	0.170284	0.000000	0.000000	0.000000	0.000000
2025-11-13	568.941529	58.632488	-0.336291	0.336291	52.752488	-3.029095	49.723392	49.723392
2025-11-14	589.482506	52.079194	-0.302235	0.302235	-6.553293	6.907748	0.354455	50.077847
2025-11-15	591.722496	51.406171	-0.298686	0.298686	-0.673023	0.677004	0.003980	50.081828
2025-11-16	601.421148	48.582505	-0.283694	0.283694	-2.823666	2.896855	0.073189	50.155017
2025-11-17	575.860407	56.346650	-0.324514	0.324514	7.764145	-7.251439	0.512706	50.667723
2025-11-18	561.484586	61.188586	-0.349331	0.349331	4.841936	-4.665161	0.176775	50.844498
2025-11-19	562.699440	60.765507	-0.347182	0.347182	-0.423080	0.424386	0.001306	50.845805
2025-11-20	558.135309	62.368575	-0.355304	0.355304	1.603068	-1.584584	0.018485	50.864289
2025-11-21	566.642594	59.410170	-0.340273	0.340273	-2.958405	3.022674	0.064269	50.928558
2025-11-22	570.093618	58.246193	-0.334309	0.334309	-1.163976	1.174290	0.010314	50.938872
2025-11-23	556.695607	62.881970	-0.357894	0.357894	4.635776	-4.479070	0.156706	51.095577
2025-11-24	561.321606	61.245544	-0.349620	0.349620	-1.636426	1.655617	0.019192	51.114769
2025-11-25	544.283908	67.464993	-0.380770	0.380770	6.219450	-5.956716	0.262734	51.377503
2025-11-26	541.793447	68.419139	-0.385477	0.385477	0.954146	-0.948292	0.005854	51.383357
2025-11-27	546.362638	66.677533	-0.376870	0.376870	-1.741606	1.761318	0.019713	51.403069
2025-11-28	533.054327	71.861035	-0.402300	0.402300	5.183502	-5.015508	0.167994	51.571063
2025-11-29	529.257371	73.402677	-0.409755	0.409755	1.541642	-1.527516	0.014126	51.585189
2025-11-30	521.803661	76.512148	-0.424641	0.424641	3.109471	-3.054197	0.055274	51.640463
2025-12-01	520.912809	76.891242	-0.426443	0.426443	0.379094	-0.378292	0.000802	51.641265
2025-12-02	510.335046	81.516438	-0.448176	0.448176	4.625196	-4.510809	0.114387	51.755652
2025-12-03	504.506401	84.164286	-0.460418	0.460418	2.647848	-2.612260	0.035587	51.791239
2025-12-04	491.525427	90.321132	-0.488319	0.488319	6.156846	-5.976676	0.180170	51.971409
2025-12-05	505.797811	83.571460	-0.457690	0.457690	-6.749672	6.969482	0.219810	52.191219
2025-12-06	507.711548	82.699417	-0.453664	0.453664	-0.872043	0.875898	0.003856	52.195075
2025-12-07	520.808538	76.935719	-0.426654	0.426654	-5.763698	5.941628	0.177929	52.373005
2025-12-08	524.530577	75.361673	-0.419157	0.419157	-1.574045	1.588022	0.013976	52.386981
2025-12-09	509.857886	81.730527	-0.449172	0.449172	6.368854	-6.150164	0.218690	52.605671
2025-12-10	501.340450	85.632580	-0.467144	0.467144	3.902053	-3.825790	0.076263	52.681933
2025-12-11	496.954040	87.702252	-0.476548	0.476548	2.069673	-2.049084	0.020589	52.702522

델타헤징 포트폴리오 시뮬레이션 (Cont'd)




```
===== Final PnL Statistics =====
Mean PnL      : 50.60
Std Dev PnL   : 12.67
Min PnL       : 14.97
Max PnL       : 80.28
```