Two Stock ELS (OS FDM Method)

20213858조규영

□ Two Stock ELS(삼성증권 제 26165회 주가연계증권)

지수/종목 연계 ELS - 만기 3년 원금비보장형

삼성증권 제26165회 주가연계증권

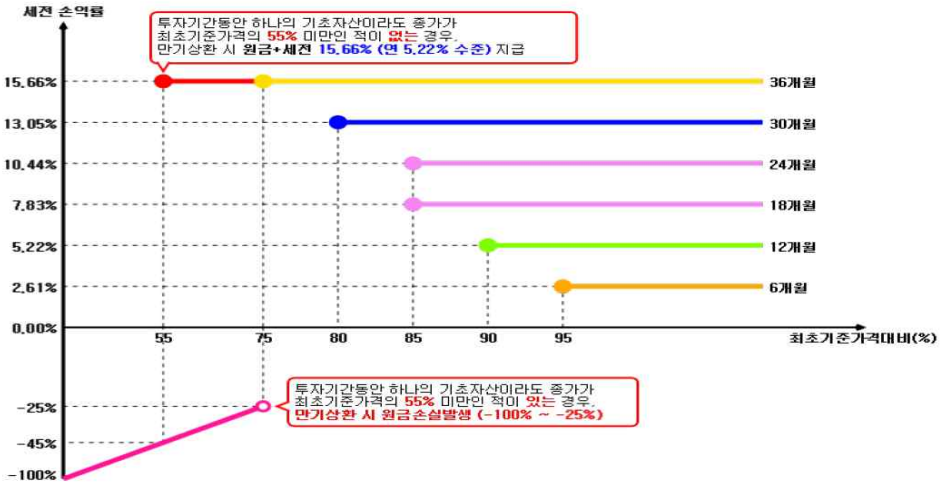
Ⅰ. 상품의 개요

상품종류	2Star Step-down 6Chance
위험등급	초고위험
고객투자성향	초고위험투자형
기초자산	S&P500, 삼성전자 보통주
판매예정한도	200억원 (예정)
최저가입금액	10만원 이상, 1만원 단위
청약기간	2021년 05월 25일(화) ~ 2021년 06월 03일(목)
만기일(예정)	2024년 06월 03일(월)

※ELS 상품 개요

- 상품명 : 삼성증권 제26165회 주가연계증권
- 기초자산 : 삼성전자(보통주), S&P500
- 상품유형 : 스텝다운 조기상환형
- 만기/상환주기 : 3년/6개월
- 상환 조건 및 하락한계가격 : 95-90-85-85-80 / 55
- 특정조건 충족시: 세전 연 5.22%
- 발행일: 2021/06/04
- 만기일: 2024/06/03

□ Payoff Structure 및 상환 조건



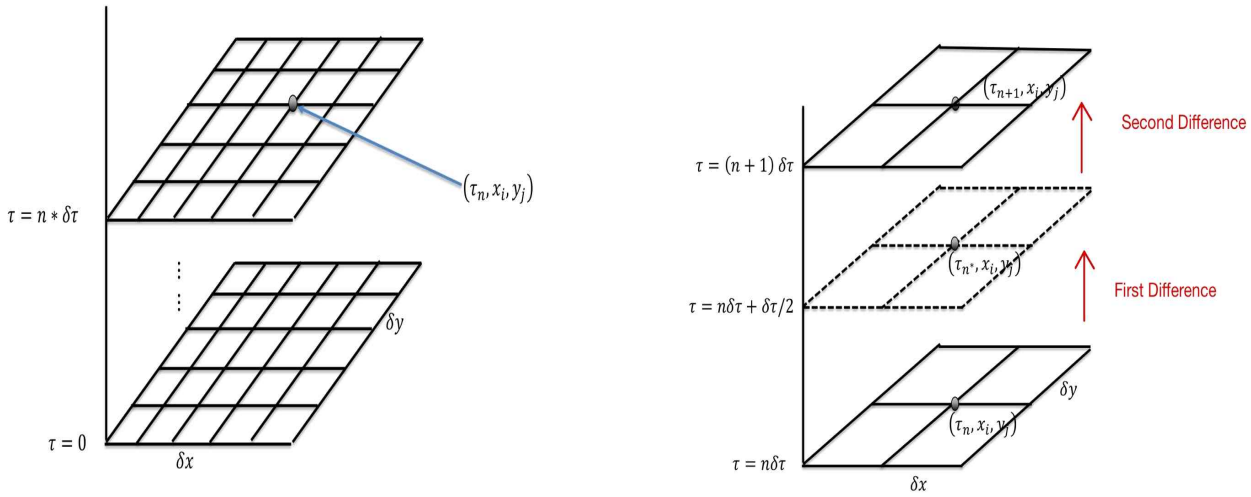
□ 예상손익구조

구분	내용	투자손익률(세전)
자동조기/만기상환	① 1차 중간기준가격 결정일에 각 기초자산의 해당 중간기준가격이 모두 최초기준가격의 95%보다 크거나 같게 되는 경우	연 5.22% 수준
	② 위 ①의 경우가 발생하지 않고, 2차 중간기준가격 결정일에 각 기초자산의 해당 중간기준가격이 모두 최초기준가격의 90%보다 크거나 같게 되는 경우	
	③ 위 ②의 경우가 발생하지 않고, 3차 중간기준가격 결정일에 각 기초자산의 해당 중간기준가격이 모두 최초기준가격의 85%보다 크거나 같게 되는 경우	
	④ 위 ③의 경우가 발생하지 않고, 4차 중간기준가격 결정일에 각 기초자산의 해당 중간기준가격이 모두 최초기준가격의 85%보다 크거나 같게 되는 경우	
	⑤ 위 ④의 경우가 발생하지 않고, 5차 중간기준가격 결정일에 각 기초자산의 해당 중간기준가격이 모두 최초기준가격의 80%보다 크거나 같게 되는 경우	
	⑥ 위 ⑤의 경우가 발생하지 않고, 최종기준가격 결정일에 각 기초자산의 해당 최종기준가격이 모두 최초기준가격의 75%보다 크거나 같게 되는 경우	
만기상환	⑦ 위 ⑥의 경우가 발생하지 않고, 최초기준가격 결정일(불포함) 이후 마지막 최종기준가격 결정일(포함) 이전에 하나의 기초자산이라도 종가에 최초기준가격의 55% 미만으로 내려간 적이 없는 경우	연 5.22% 수준
	⑧ 위 ⑥의 경우가 발생하지 않고, 최초기준가격 결정일(불포함) 이후 마지막 최종기준가격 결정일(포함) 이전에 하나의 기초자산이라도 종가에 최초기준가격의 55% 미만으로 내려간 적이 있는 경우	-100% ~ -25%

☐ Properties of Operator Splitting FDM

- Implicit scheme in x variable only in the first time lag
- Implicit Scheme in y variable only in the second time lag
- No explicit schemes

☐ Model Structure(Operator Splitting FDM)



Step1) Black Scholes Equation (Quanto ELS)

$X(t)$: $KOSPI200's\ value\ at\ a\ time\ t$
$Y(t)$: $SP500's\ value\ at\ a\ time\ t$
$u(\tau, x, y)$: $the\ derivative\ value$ and $x = X(t),\ y = Y(t),\ \tau = T - t$

$$\frac{\partial u}{\partial \tau} = (r - q_x)x\frac{\partial u}{\partial x} + (r - q_y - \rho_F \sigma_F \sigma_y)y\frac{\partial u}{\partial y} + \frac{\sigma_1^2 x^2}{2}\frac{\partial^2 u}{\partial x^2} + \frac{\sigma_2^2 y^2}{2}\frac{\partial^2 u}{\partial y^2} + \rho\sigma_1\sigma_2 xy\frac{\partial u^2}{\partial x \partial y} - ru$$

Step2) Time steps

$$\frac{u_{i.j}^{n+1}-u_{i,j}^{n}}{dt} = \frac{u_{i,j}^{n^*}-u_{i,j}^{n}}{dt} + \frac{u_{i,j}^{n+1}-u_{i,j}^{n^*}}{dt} \equiv \mathcal{L}_{OS}^{x} u_{i,j}^{n^*} + \mathcal{L}_{OS}^{y} u_{i,j}^{n}$$

$$\frac{u_{i,j}^{n^*}-u_{i,j}^{n}}{dt} \equiv \mathcal{L}_{OS}^{x} u_{i,j}^{n^*} (implicit\ x-direction)$$

$$\frac{u_{i,j}^{n+1}-u_{i,j}^{n^*}}{dt} \equiv \mathcal{L}_{OS}^{y} u_{i,j}^{n} (implicit\ y-direction)$$

$$n^* : auxiliary\ \ timestep,\ n^*\partial\tau = \tau + \partial\tau/2$$

Step3) Difference Scheme & Boundary Condition

Difference Scheme

$$\mathcal{L}_{OS}^{x} u_{i,j}^{n^*} = (r-q_x)x_i \frac{u_{i+1,j}^{n^*}-u_{i,j}^{n}}{dt} + (\sigma_x x_i)^2 \frac{u_{i+1,j}^{n^*}-2u_{i,j}^{n^*}+u_{i-1,j}^{n^*}}{2(dt)^2} + \frac{\rho\sigma_x\sigma_y x_i y_i}{2} \frac{u_{i+1,j+1}^{n}+u_{i-1,j-1}^{n}-u_{i-1,j+1}^{n}-u_{i+1,j-1}^{n}}{4(dt)^2} - \frac{r}{2}u_{i,j}^{n^*}$$

$$\mathcal{L}_{OS}^{y} u_{i,j}^{n+1} = (r-q_y-\rho_F\sigma_F\sigma_y)y_i \frac{u_{i,j+1}^{n+1}-u_{i,j}^{n+1}}{dt} + (\sigma_y y_i)^2 \frac{u_{i,j+1}^{n+1}-2u_{i,j}^{n+1}+u_{i,j-1}^{n+1}}{2(dt)^2} + \frac{\rho\sigma_x\sigma_y x_i y_i}{2} \frac{u_{i+1,j+1}^{n^*}+u_{i-1,j-1}^{n^*}-u_{i-1,j+1}^{n^*}-u_{i+1,j-1}^{n^*}}{4(dt)^2} - \frac{r}{2}u_{i,j}^{n+1}$$

Linear bounary conditions

$$u(\tau,0,y) = u_{0,j}^{n} = 2u_{1,j}^{n} - u_{2,j}^{n}, \quad u(\tau,x_{\max},y) = u_{N_x,j}^{n} = 2u_{N_x-1,j}^{n} - u_{N_x-2,j}^{n}$$

$$u(\tau,x,0) = u_{0,j}^{n} = 2u_{1,j}^{n} - u_{2,j}^{n}, \quad u(\tau,x,y_{\max}) = u_{j,N_y}^{n} = 2u_{j,N_y-1}^{n} - u_{j,N_y-2}^{n}$$

Step4) Difference Equation

The First Difference Equation

$$b_i u_{i-1,j}^{n^*} + a_i u_{i,j}^{n^*} + c_i u_{i+1,j}^{n^*} = d_n$$

$$a_i = \frac{1}{dt} + \frac{(r-q_x)x_i}{dx} + \frac{(\sigma_x x_i)^2}{(dx)^2} + \frac{r}{2}$$

$$b_i = -\frac{(\sigma_x x_i)^2}{2(dx)^2}$$

$$c_i = -\frac{(\sigma_x x_i)^2}{2(dx)^2} - \frac{(r-q_x)x_i}{dx}$$

$$d_i^n = \frac{u_{i,j}^n}{dt} + \frac{\rho\sigma_x\sigma_y x_i y_i}{2 \cdot 4(dx)(dy)}(u_{i+1,j+1}^n + u_{i-1,j-1}^n - u_{i-1,j+1}^n - u_{i+1,j-1}^n)$$

The Second Difference Equation

$$b_j u_{i-1,j}^{n+1} + a_j u_{i,j}^{n+1} + c_j u_{i+1,j}^{n+1} = d_{n^*}$$

$$a_j = \frac{1}{dt} + \frac{(r-q_y-\rho_F\sigma_F\sigma_y)y_j}{dy} + \frac{(\sigma_y y_j)^2}{(dy)^2} + \frac{r}{2}$$

$$b_j = -\frac{(\sigma_y y_j)^2}{2(dy)^2}$$

$$c_j = -\frac{(\sigma_y y_j)^2}{2(dy)^2} - \frac{(r-q_y-\rho_F\sigma_F\sigma_y)y_j}{dy}$$

$$d_j^{n^*} = \frac{u_{i,j}^{n^*}}{dt} + \frac{\rho\sigma_x\sigma_y x_i y_i}{2 \cdot 4(dx)(dy)}(u_{i+1,j+1}^{n^*} + u_{i-1,j-1}^{n^*} - u_{i-1,j+1}^{n^*} - u_{i+1,j-1}^{n^*})$$

□ Parameter

삼성전자 보통주(005930)	82,800원	2021/06/03(종가)
삼성전자's volatility	27.81%	6개월 변동성과 3년 변동성의 평균
삼성전자's dividend	3.7%	2020년 배당률
S&P500	4,192.85	2021/06/03(종가)
S&P500's volatility	22.25%	6개월 변동성과 3년 변동성의 평균
S&P500's dividend	1.94%	2020년 배당률
Correlation(KOSPI & S&P)	0.23680	6개월 상관계수와 3년 상관계수의 평균
Risk-Free Rate	1.0780%	1년 국채
Volatility of percentage change of exchange rate	7.64%	KRW/USD의 1년 변동성(Historical)
Correlation between S&P500 and exchange rate	0.58915	KRW/USD와 S&P500의 1년 상관계수(Historical)

□ Parameters's properties

-Underlying Asset & Asset's volatility

기초자산의 가격이 하락하면 ELS 현재가치가 하락한다. 또한 듀레이션이 길어져 hedging cost가 증가한다.
또한 기초자산의 변동성이 증가하면 ELS가치는 하락하고 발행자에게 이익이다.

-Risk-free rate

무위험 이자율이 상승하면 drift가 이론상 상승하고 발행자에게 손실을 입히지만 ELS 가격에 크게 민감하지 않다.

-Dividend

배당금이 커지면 drift term이 감소하므로 ESL 가치가 감소한다.

-correlation

만약 correlation이 음수이면 한 주가가 상승할 때, 다른 주가는 하락할 가능성이 증가하게 된다. 따라서 barrier를 치게 될 가능성이 커지므로 ELS 가치는 하락하고 발행자에게 이익이 된다. 반대로 correlation이 강한 양수일 때, 두 개의 주가는 같은 방향으로 움직일 가능성이 크고 만약 둘 다 하락한다면 hedging instrument의 가격도 하락할 가능성이 크다. 따라서 발행자 입장에서는 손실을 입을 수 있다.

-Quanto correlation

일반적으로 음수가 되므로 S&P500에 대한 drift term이 증가하게 되므로 ELS 가치가 상승하고 이는 발행자에게 손실이 된다.

-Quanto volatility

일반적으로 S&P500의 변동성이 증가하면 Quanto volatility도 증가한다. 따라서 Quanto correlation이 큰 음수가 될 것이고 이는 발행자에게 손실을 가져오게 된다.

ELS Price(MonteCarlo Simulation) 중 일부 코드

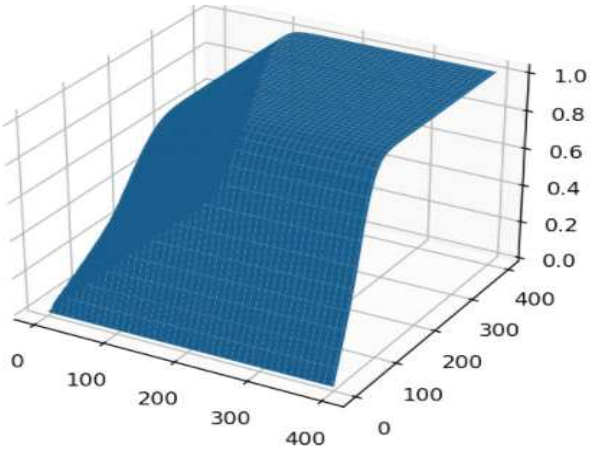
```
for j in range(n):
    w0 = np.random.normal(0,1,size=[tot_date,2])
    w0 = np.transpose(w0)

    #상관관계가 있는 난수 생성
    w  = np.dot(k,w0)            # k = np.linalg.cholesky(corr)

    payoff=np.zeros([repay_n,1])

    for j in range(tot_date):
        #total 기간동안의 몬테카를로 시뮬레이션
        S1[j+1] =S1[j]*np.exp((r-0.5*x_vol**2)*dt+x_vol*np.sqrt(dt)*w[0,j])
        S2[j+1] =S2[j]*np.exp((r-0.5*y_vol**2)*dt+y_vol*np.sqrt(dt)*w[1,j])

    R1 = S1/ratio_S1
    R2 = S2/ratio_S2                                  # 비율로 변환

    WP = np.minimum(R1,R2)                            #각각의 j항에서 최소의 값의 array
    WP_checkday=WP[check_day]                         #check_day에서의 최소 값들

    repay_event = 0

    for j in range(repay_n):                           # repay_n = 조기상환횟수 6

        if WP_checkday[j] >= strike_price[j]:
            payoff[j] = payment[j]
            repay_event = 1
            num[j] = num[j] + 1
            break

    if repay_event == 0:
        if min(WP) > kib :

            payoff[-1] = face_value*(1+dummy)
            num[-1] = num[-1] + 1
        else:

            payoff[-1]= face_value*WP[-1]
            num[-1] = num[-1] + 1

    tot_payoff=tot_payoff + payoff

# 시뮬레이션의 평균
mean_payoff = tot_payoff/n

#무위험이자율로 할인

for j in range(repay_n):
    discount_payoff[j] = mean_payoff[j] * np.exp(-r*check_day[j]/oneyear)

#ELS 공정가격

price = np.sum(discount_payoff)
```

ELS Price(OS FDM) 중 일부 코드

```
for num_dt in range(Nt):

    if (num_dt-1) in redem_idx :
        for (tmp_idx, temp_num) in enumerate(redem_idx[:-1]) :
            if int(num_dt-1) == temp_num :
                    for i in range(0,Nx+1):
                        for j in range(0,Ny+1) :
                            if min(x[i]/S1, y[j]/S2) >= B :
                                u0[i,j] = 1 + cr[5-(tmp_idx+1)]
                            else:
                                u0[i,j] = min(x[i]/S1, y[j]/S2)
                            if min(x[i]/S1, y[j]/S2) >= K[5-(tmp_idx+1)] :
                                w0[i,j] = 1 + cr[5-(tmp_idx+1)]
                            else:
                                w0[i,j] = min(x[i]/S1, y[j]/S2)
    for num_dy in range(1,Ny):
        for num_dx in range(1,Nx):
            d_ux[num_dy-1,num_dx-1] = (
               (1/2)*rho*sig1*sig2*x[num_dx]*y[num_dy]
                *(u0[num_dx+1,num_dy+1] - u0[num_dx+1,num_dy-1] - u0[num_dx-1,num_dy+1] + u0[num_dx-1,num_dy-1])
                /(4*dx*dy) + u0[num_dx,num_dy]/dt
            )
            d_wx[num_dy-1,num_dx-1] = (
               (1/2)*rho*sig1*sig2*x[num_dx]*y[num_dy]
                *(w0[num_dx+1,num_dy+1] - w0[num_dx+1,num_dy-1] - w0[num_dx-1,num_dy+1] + w0[num_dx-1,num_dy-1])
                /(4*dx*dy) + w0[num_dx,num_dy]/dt
            )
    u[1:Nx,1:Ny] = thomas_mat1 @ d_ux
    w[1:Nx,1:Ny] = thomas_mat1 @ d_wx

    u[0,1:] = 2*u[1,1:] - u[2,1:]
    u[-1,1:] = 2*u[-2,1:] - u[-3,1:]
    u[:,0] = 2*u[:,1] - u[:,2]
    u[:,-1]= 2*u[:,-2] - u[:,-3]
    w[0,1:] = 2*w[1,1:] - w[2,1:]
    w[-1,1:] = 2*w[-2,1:] - w[-3,1:]
    w[:,0] = 2*w[:,1] - w[:,2]
    w[:,-1]= 2*w[:,-2] - w[:,-3]

    w0 = w.copy()
    u0 = u.copy()

    for num_dx in range(1,Nx):
        for num_dy in range(1,Ny):
            d_uy[num_dx-1,num_dy-1] = (
                (1/2)*rho*sig1*sig2*x[num_dx]*y[num_dy]
                *(u0[num_dx+1,num_dy+1] - u0[num_dx+1,num_dy-1] - u0[num_dx-1,num_dy+1] + u0[num_dx-1,num_dy-1])
                /(4*dx*dy) + u0[num_dx,num_dy]/dt
            )
            d_wy[num_dx-1,num_dy-1] = (
                (1/2)*rho*sig1*sig2*x[num_dx]*y[num_dy]
                *(w0[num_dx+1,num_dy+1] - w0[num_dx+1,num_dy-1] - w0[num_dx-1,num_dy+1] + w0[num_dx-1,num_dy-1])
                /(4*dx*dy) + w0[num_dx,num_dy]/dt
            )

    u[1:Nx,1:Ny] = (thomas_mat2 @ d_uy.T).T
    w[1:Nx,1:Ny] = (thomas_mat2 @ d_wy.T).T

    u[:,1:int(100*B)+1] = w[:,1:int(100*B)+1].copy()
    u[1:int(100*B)+1,:] = w[1:int(100*B)+1,:].copy()

    u[0,1:] = 2*u[1,1:] - u[2,1:]
    u[-1,1:] = 2*u[-2,1:] - u[-3,1:]
    u[:,0] = 2*u[:,1] - u[:,2]
    u[:,-1]= 2*u[:,-2] - u[:,-3]
    w[0,1:] = 2*w[1,1:] - w[2,1:]
    w[-1,1:] = 2*w[-2,1:] - w[-3,1:]
    w[:,0] = 2*w[:,1] - w[:,2]
    w[:,-1]= 2*w[:,-2] - w[:,-3]

    w0 = w.copy()
    u0 = u.copy()
```

□Compare Simulated price and FDM price with the price the bank suggests.

Bank Suggests	8,797 원
MonteCarlo Simulation	9,103 원
Operating splitting(OS) FDM	8,966 원



※OS FDM Price Space(3D)

□ The causes of the price difference.

1. Error of Estimate input value

기초자산을 평가할 때, 배당금, 변동성, 상관계수를 역사적 데이터를 사용해서 추정한 값을 넣었기 때문에 Input parameter의 추정오차가 발생했고 이로 인해 가격차이가 발생하고 잘못된 공정한 가격을 추정한 원인입니다.

2. Assumption of Volatility

위의 모델은 Black-Scholes-Equation을 통한 OS-FDM을 이용해 가격을 추정하였습니다. 이때, 각 기초자산의 변동성이 변수가 아닌 상수라고 가정을 합니다. 그러나 실제 현실에서는 각 parameter들이 상수가 아닌 시간에 따라 변하는 변수이기 때문에 정확한 가격을 추정하기 위해서는 시간이 변함에 따라 parameter들의 변화를 고려해야합니다. 위의 모델에서는 이를 고려하지 못해서 잘못된 공정가격을 추정하는 원인입니다.

3. The limit of the FDM model

FDM은 Taylor' Remainder Theorem을 이용해서 가격을 추정하는 방식입니다.
이때, 계산 식에서 오차항이 발생하게 되는데 이로인해 추정오차가 발생하게 됩니다.
따라서 Taylor' Remainder Theorem로 인해 공정가격의 오차를 발생시키게 됩니다.

□ Greeks(모든 기준은 t=0 시점의 주가, ELS가격은 1을 기준으로 하였음)

□ Delta

정의 : 기초자산의 가격 변화에 대한 파생상품 가격의 변화

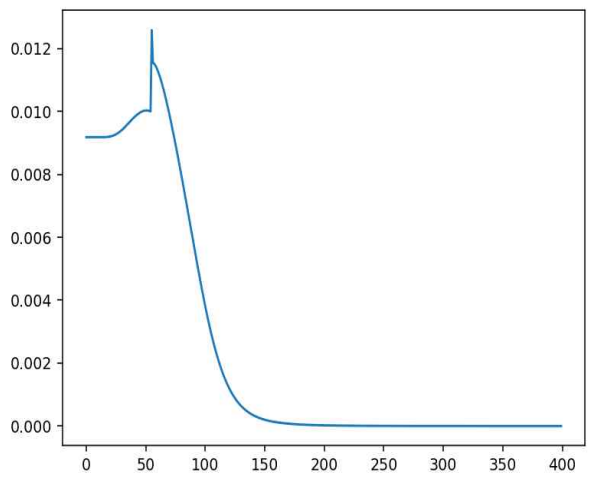
$$\frac{u(\tau, x+\partial x, y) - u(\tau, x, y)}{\partial x} = \Delta_x$$

$$\frac{u(\tau, x, y+\partial y) - u(\tau, x, y)}{\partial y} = \Delta_y$$

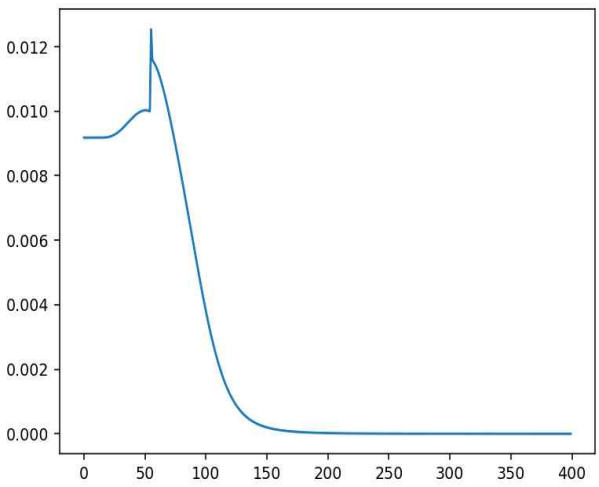
Δ_x : 삼성전자 주가가 1%(약 828원) 상승하면 ELS의 가격은 대략 39.84원 움직인다.

Δ_y : $SP500$ 지수가 1%(약 41.9) 상승하면 ELS의 가격은 대략 39.79원 움직인다.

※삼성전자(Delta = 0.003984)



※S&P500 (Delta= 0.003979)



□ Gamma

정의 : 델타 변화에 대한 파생상품 가격의 변화

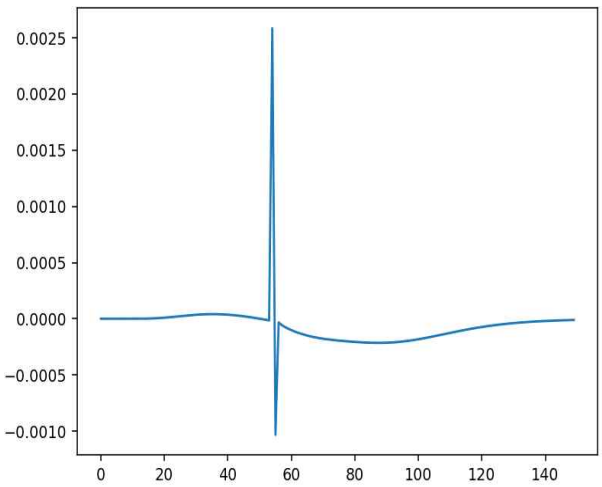
$$\frac{\Delta_x}{\partial x} = \frac{u(\tau, x+\partial x, y) - 2u(\tau, x, y) + u(\tau, x-\partial x, y)}{\partial x^2} = \Gamma_x$$

$$\frac{\Delta_y}{\partial y} = \frac{u(\tau, x, y+\partial y) - 2u(\tau, x, y) + u(\tau, x, y-\partial y)}{\partial y^2} = \Gamma_y$$

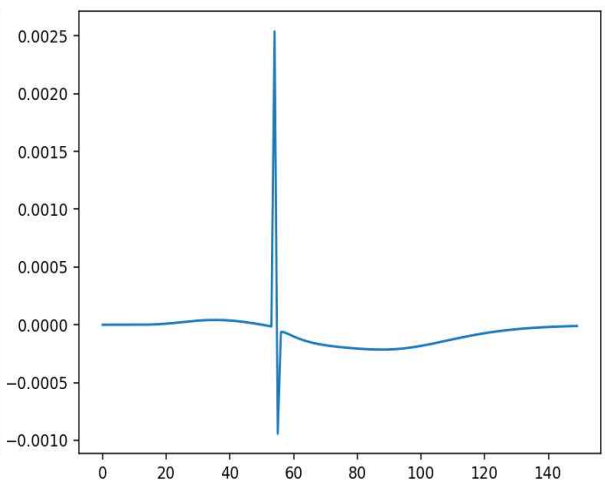
Γ_x : 삼성전자 주가가 1%(약 828원) 상승하면 델타는 대략 -0.000188 움직인다.

Γ_y : $SP500$ 지수가 1%(약 41.9) 상승하면 델타는 대략 -0.000188 움직인다.

※삼성전자(Gamma = -0.000188)



※S&P500 (Gamma = -0.000188)



□ Vega

정의 : 변동성 변화에 대한 파생상품 가격의 변화

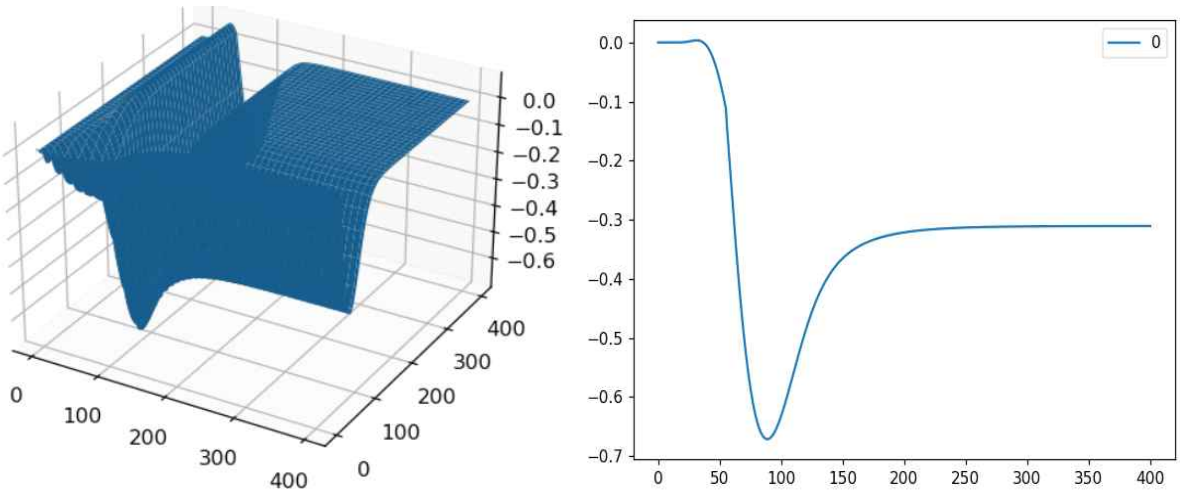
$$\frac{u(\tau,x,y)^{*}-u(\tau,x,y)}{\partial\sigma_{x}}=\nu_{x}$$

$$\frac{u(\tau,x,y)^{*}-u(\tau,x,y)}{\partial\sigma_{y}}=\nu_{y}$$

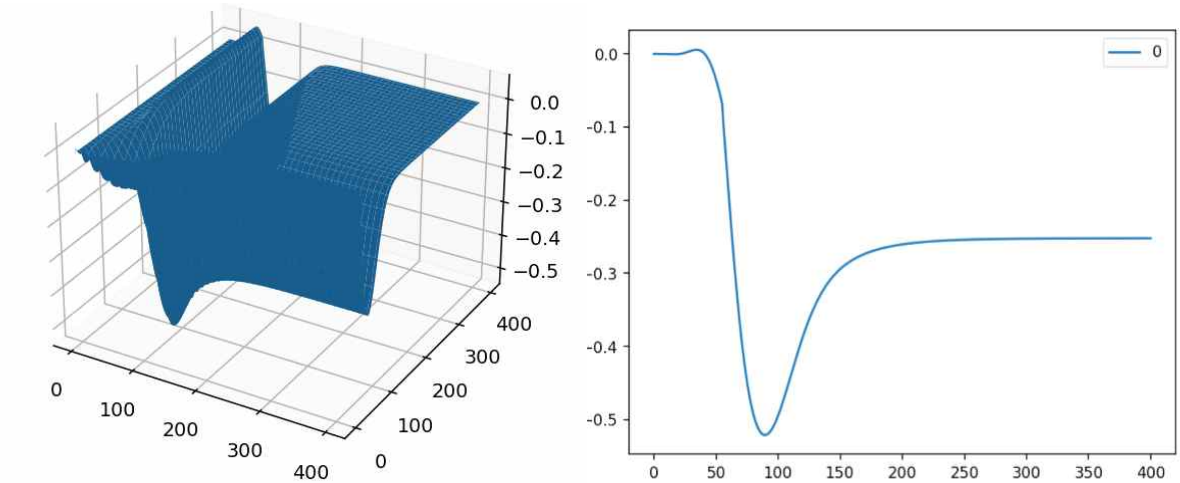
ν_x : 삼성전자 주가의 변동성이 1%(약 0.2781%) 움직이면 ELS의 가격은 대략 -17.0726원 움직인다.

ν_y : $SP500$ 지수의 변동성이 1%(약 0.2225%) 움직이면 ELS의 가격은 대략 -11.0032원 움직인다.

※삼성전자(Vega= -0.613902)



※S&P500 (Vega= -0.494527)



□ Rho(=0.896636)

정의 : 이자율 변화에 대한 파생상품 가격의 변화

$$\frac{u(\tau,x,y)^{*}-u(\tau,x,y)}{\partial RF}=\rho_x$$

ρ_x : 무위험 이자율이 $1bp(0.01\%)$ 움직이면 ELS의 가격은는 대략 0.8966원 움직인다.