IRSwaption 평가 설명서

1. Black Formula for Swaption

Swaption의 기초자산을 F(=forward swap rate)라고 하자.

$$F(0,\mathrm{T_{start}},\mathrm{T_{end}}) = \frac{\mathrm{P}(0,\mathrm{T_{start}}) - P(0,T_{end})}{\sum_{i=1}^{N}\Delta(T_{i-1},T_i)P(0,T_i)}$$

$$P(Payer\ \ Swaption) = N \times Annuity \times [f_0(T_{start},T_{end})N(d_1) - XN(d_2)]$$

$$\mathrm{N(y)} = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{y} e^{-\frac{z^2}{2}}dz$$

$$\mathrm{d}_1 = \frac{\left(\ln\left(\frac{f_0(T_{start},T_{end})}{X}\right) + \frac{1}{2}v^2T\right)}{v\sqrt{T}} \quad ,\mathrm{d}_2 = d_1 - v\sqrt{T}$$

2. Bermudan Swaption

HW 1F FDM을 통해 계산

Discount Factor

$$\mathrm{P(t,T)} = \hat{E}\left(\mathrm{e}^{-\int_{\mathrm{t}}^{\mathrm{T}} r(u)du}\right)$$

The instantaneous spot rate, called "short rate", is the interest rate r(t).

HW 1F Dynamics of Short Rate

$$\mathrm{dr(t)} = [\theta(\mathrm{t}) - \kappa \cdot r(t)]dt + \sigma(t)dW = \kappa\left[\frac{\theta(\mathrm{t})}{\kappa} - r(t)\right]dt + \sigma(t)dW$$

if $r(t) > \frac{\theta(\mathrm{t})}{\kappa}$ Then Drift have (-) else (+)

$$\mathrm{r(t)} = \alpha(\mathrm{t}) + x(t)$$

$$\mathrm{d}\alpha(\mathrm{t}) = [\theta(t) - \kappa\alpha(t)]dt$$

$$\mathrm{d}x(t) = -\kappa x(t)dt + \sigma dW$$

HW 1F PDE

$$\mathrm{U_t} = \kappa x \mathrm{U}_x - \frac{1}{2}\sigma^2 \mathrm{U}_{xx} + [\alpha(t) + x]\mathrm{U}$$

$$\alpha(\mathrm{t}) = -\frac{\delta P(0,t)}{\delta t} + \frac{\sigma^2}{2\kappa^2}(1 - e^{-\kappa t})^2$$

Implicit Finite Difference Method

$$\frac{\mathrm{u_i^{k+1} - u_i^k}}{\mathrm{h_t}} = \kappa \mathrm{x} \cdot \frac{\left(\mathrm{u_{i+1}^k - u_{i-1}^k}\right)}{2\mathrm{h_x}} - \frac{1}{2}\sigma^2 \frac{\left(\mathrm{u_{i+1}^k + u_{i-1}^k - 2u_i^k}\right)}{\mathrm{h_x^2}} + [\alpha(t_{k+1}) + \mathrm{x}_i]\mathrm{u_i^{k+1}}$$

$$\mathrm{u_i^{k+1} - u_i^k} = \mathrm{h_t}\kappa \mathrm{x} \cdot \frac{\left(\mathrm{u_{i+1}^k - u_{i-1}^k}\right)}{2\mathrm{h_x}} - \frac{\mathrm{h_t}}{2}\sigma^2 \frac{\left(\mathrm{u_{i+1}^k + u_{i-1}^k - 2u_i^k}\right)}{\mathrm{h_x^2}} + \mathrm{h_t}[\alpha(t_{k+1}) + \mathrm{x}_i]\mathrm{u_i^{k+1}}$$

$$\mathrm{u_i^{k+1}}(1 - h_t[\alpha(t_{k+1}) + \mathrm{x}_i]) = \left[-\frac{\mathrm{h_t}\kappa x_i}{2h_x} - \frac{h_t\sigma^2}{2h_x^2}\right]u_{i-1}^k + \left[1 + \frac{\mathrm{h_t}\sigma^2}{\mathrm{h_x^2}}\right]u_i^k + \left[\frac{\mathrm{h_t}\kappa x_i}{2h_x} - \frac{h_t\sigma^2}{2h_x^2}\right]u_{i+1}^k$$

$$\mathrm{u_i^{k+1}}W_i = A_i u_{i-1}^k + B u_i^k + C_i u_{i+1}^k$$

$$\mathrm{W_i} = (1 - h_t[\alpha(t_{k+1}) + \mathrm{x}_i])$$

$$A_i = \left[-\frac{\mathrm{h_t}\kappa x_i}{2h_x} - \frac{h_t\sigma^2}{2h_x^2}\right]$$

$$B = \left[1 + \frac{\mathrm{h_t}\sigma^2}{\mathrm{h_x^2}}\right]$$

$$C_i = \left[\frac{\mathrm{h_t}\kappa x_i}{2h_x} - \frac{h_t\sigma^2}{2h_x^2}\right]$$

Boundary Condition

$u_{-1} = 2u_0 - u_1$

$u_{N+1} = 2u_N - u_{N-1}$

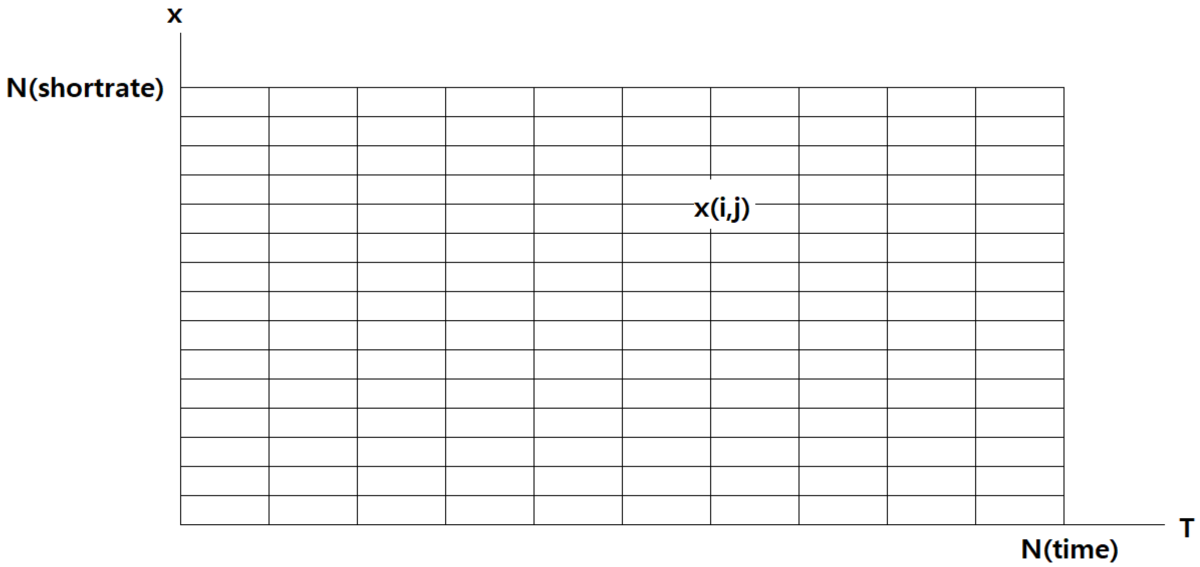
$\mathrm{u_0^{k+1}}W_0 = A_0 u_{-1}^k + B u_0^k + C_i u_1^k = A_0(2u_0^k - u_1^k) + B u_0^k + C_0 u_1^k$

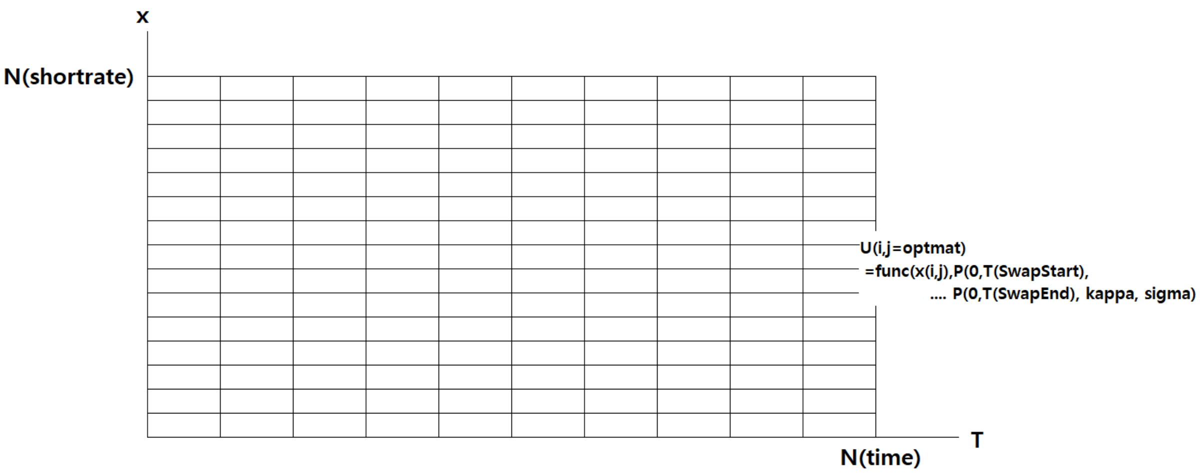
$\mathrm{u_0^{k+1}}W_0 = u_0^k(2A_0 + B) + u_1^k(C_0 - A_0)$

$\mathrm{u_N^{k+1}}W_N = A_N u_{N-1}^k + B u_N^k + C_i u_{N+1}^k = A_N u_{N-1}^k + B u_N^k + C_N\left(2u_N^k - u_{N-1}^k\right)$

$\mathrm{u_N^{k+1}}W_N = \mathrm{u_{N-1}^k}(A_N - C_N) + u_N^k(B + 2C_N)$

$$
\begin{bmatrix}
2A_0+B & C_0-A_0 & 0 & 0 & \cdots & \cdots & 0 & 0 & 0 \\
A_1 & B & C_1 & 0 & \cdots & \cdots & 0 & 0 & 0 \\
0 & A_2 & B & C_2 & \cdots & \cdots & 0 & 0 & 0 \\
0 & 0 & A_3 & B & \cdots & \cdots & 0 & 0 & 0 \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 0 & 0 \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 0 & 0 \\
\cdots & \cdots & \cdots & 0 & 0 & A_{N-2} & B & C_{N-2} & 0 \\
\cdots & \cdots & \cdots & 0 & 0 & 0 & A_{N-1} & B & C_{N-1} \\
\cdots & \cdots & \cdots & 0 & 0 & 0 & 0 & A_N-C_N & B+2C_N
\end{bmatrix}
\begin{bmatrix}
u_0^k \\ u_1^k \\ u_2^k \\ u_3^k \\ \cdots \\ \cdots \\ u_{N-2}^k \\ u_{N-1}^k \\ u_N^k
\end{bmatrix}
=
\begin{bmatrix}
W_0u_0^{k+1} \\ W_1u_1^{k+1} \\ W_2u_2^{k+1} \\ W_3u_3^{k+1} \\ \cdots \\ \cdots \\ W_{N-2}u_{N-2}^{k+1} \\ W_{N-1}u_{N-1}^{k+1} \\ W_Nu_N^{k+1}
\end{bmatrix}
$$





HW Forward Discount Factor

$$\mathrm{P}(\mathrm{t},\mathrm{T}) = \hat{E}\left(\mathrm{e}^{-\int_{\mathrm{t}}^{\mathrm{T}} r(u)du}\right) = \hat{E}\left(\mathrm{e}^{-\int_{\mathrm{t}}^{\mathrm{T}}(x(u)+\alpha(u))du}\right)$$

여기서 $\int_{\mathrm{t}}^{\mathrm{T}} x(u)du$는 평균 x(t)B(t,T), 분산 V(t,T)인 정규분포를 따른다.

$$\mathrm{B}(\mathrm{t},\mathrm{T}) = \frac{1-e^{-\kappa(T-t)}}{\kappa}$$

$$\mathrm{V}(\mathrm{t},\mathrm{T}) = \int_{\mathrm{t}}^{\mathrm{T}} \sigma^2(u)B^2(u,T)du \approx \int_{\mathrm{t}}^{\mathrm{T}} \frac{\sigma^2\left[1-2e^{-\kappa(T-u)}+e^{-2\kappa(T-u)}\right]}{\kappa^2}du$$

$$= \frac{\sigma^2}{\kappa^2}\left(\mathrm{T}-\mathrm{t}+2\frac{\mathrm{e}^{-\kappa(\mathrm{T}-\mathrm{t})}-1}{\kappa} - \frac{\mathrm{e}^{-2\kappa(\mathrm{T}-\mathrm{t})}-1}{2\kappa}\right)$$

따라서 위험 중립 측도에서 $\mathrm{P}(\mathrm{t},\mathrm{T})$는 정규분포의 적률생성함수(Moment Generate Function)에 따라 다음과 같다.

※ $\mathrm{E}(\mathrm{e}^{\mathrm{tx}}) = e^{\mu t+\frac{1}{2}(\sigma t)^2}$ MGF of Normal Distribution 따라서,

$$\mathrm{P}(\mathrm{t},\mathrm{T}) = \hat{E}\left(\mathrm{e}^{-\int_{\mathrm{t}}^{\mathrm{T}}\alpha(u)du - x(t)B(t,T)+\frac{1}{2}V(t,T)}\right)$$

0시점 시장에서 관측된 만기 T인 Zero Bond $P^{\mathrm{M}}(0,T)$가 다음을 만족한다.

$$\mathrm{P}^{\mathrm{M}}(0,T) = \mathrm{e}^{-\int_0^{\mathrm{T}}\alpha(u)du+\frac{1}{2}V(0,T)}$$

$$\mathrm{e}^{-\int_0^{\mathrm{T}}\alpha(u)du} = \mathrm{P}^{\mathrm{M}}(0,T)\mathrm{e}^{-\frac{1}{2}V(0,T)}$$

따라서

$$\mathrm{e}^{-\int_{\mathrm{t}}^{\mathrm{T}}\alpha(u)du} = \frac{P^{\mathrm{M}}(0,T)}{P^{M}(0,t)}e^{-\frac{1}{2}(V(0,T)-V(0,t))}$$

$$\mathrm{P}_{\mathrm{HW}}^{x_i}(\mathrm{t},\mathrm{T}) = \frac{P^{\mathrm{M}}(0,T)}{P^{M}(0,t)}\exp\left(-x_iB(t,T)+\frac{1}{2}\big(V(t,T)-V(0,T)+V(0,t)\big)\right)$$

$$\mathrm{P}_{\mathrm{HW}}^{x_i}(\mathrm{t},\mathrm{T}) = \frac{P^{\mathrm{M}}(0,T)}{P^{M}(0,t)}\exp\big(-x_i \times B(t,T)+QVTerm(t,T)\big)$$

HW Forward Swap Rate

$$\mathrm{FSR}_{\mathrm{HW}}^{x_i}(\mathrm{t_{timenode}}, \mathrm{t_{forwardstart}}, \mathrm{T_{swapmat}}) = \frac{P_{HW}^{x_i}(t_{timenode}, t_{forwardstart}) - P_{HW}^{x_i}(t_{timenode}, T_N)}{\sum_1^N [\Delta T_i \times P_{HW}^{x_i}(t_{timenode}, T_k)]}$$

$$* \quad \mathrm{t_{forwardstart}} = \mathrm{Forward\ \ Swap\ \ Start\ \ T}$$

$$* \quad \mathrm{T_1} = \mathrm{First\ \ Swap\ \ Coupon\ \ T}$$

$$* \quad \mathrm{T_N} = T_{\mathrm{swapmat}}$$

HW Swaption Value

- Swaption Value on node $\mathrm{x_i}$

$$\mathrm{U}(\mathrm{t_{timenode}}, \mathrm{x_i}) = max(\mathrm{FSR}_{\mathrm{HW}}^{x_i}(\mathrm{t_{timenode}}, \mathrm{t_{forwardstart}}, \mathrm{T_{swapmat}}) - Strike, 0) \sum_1^N [\Delta T_i \times P_{HW}^{x_i}(t_{timenode}, T_i)]$$

그리드 세팅 전에 미리 만들어야 할 변수들 3D Array

$$\{B(t, T_k) | k = 0 \ \ to \ \ N\} \rightarrow NOption \times Time \ \ Greed \text{개수만큼}$$

$$\{QVTerm(t, T_k) | k = 0 \ \ to \ \ N\} \rightarrow NOption \times Time \ \ Greed \text{개수만큼}$$

$$\{P(0, T_k) | k = 0 \ \ to \ \ N\} \rightarrow NOption \times Time \ \ Greed \text{개수만큼}$$

C언어 함수

1. $\mathrm{B(t,T)} = \frac{1-e^{-\kappa(T-t)}}{\kappa}$

```
double Calc_B_s_t(double kappa, double t, double s)
{
    return (1.0 - exp(-kappa * (t - s))) / kappa;
}
```

2. $\mathrm{V(t,T)} \approx \frac{\sigma^2}{\kappa^2}\left(\mathrm{T-t+2}\frac{\mathrm{e^{-\kappa(T-t)}-1}}{\kappa} - \frac{\mathrm{e^{-2\kappa(T-t)}-1}}{2\kappa}\right)$

```
double V_t_T(
    double kappa,
    double kappa2,
    double t,
    double T,
    double vol,
    double vol2
)
{
    return vol * vol2 / (kappa * kappa2) * (T - t + (exp(-kappa * (T - t)) - 1.0) / kappa + (exp(-kappa2 * (T - t)) - 1.0) / kappa2 - (exp(-(kappa + kappa2) * (T - t)) - 1.0) / (kappa + kappa2));
}

double Calc_QVTerm(double kappa, double t, double s, double sigma)
{
    return 0.5 * (V_t_T(kappa, kappa, s, t, sigma, sigma) - V_t_T(kappa, kappa, 0, t, sigma, sigma) + V_t_T(kappa, kappa, 0, s, sigma, sigma));
}
```

3. $BSSwaption = N \times Annuity \times [f_0(T_{start}, T_{end})N(d_1) - XN(d_2)]$

```
double BS_Swaption(
    long FixedPayer,                    // Fixed Payer여부
    long PriceDate,                     // 평가일
    long StartDate,                     // 스왑시작일
    long NCpn,                          // 쿠폰개수
    long* SwapDate,                     // SwapDate Array YYYYMMDD
    double NA,                          // Notional Amount
    double Vol,                         // Volatility
    double StrikePrice,                 // 행사가격
    double* Term,                       // Zero Term Structure의 Term
    double* Rate,                       // Zero Term Structure의 Rate
    long NTerm,                         // Zero Term Structure의 길이
    long DayCountFracFlag,              // DayCountFraction
    long VolFlag,                       // 0 Black Vol 1 바실리에 Normal Vol
    long PricingOrValueFlag,            // Pricing할지 Valuation할지
    double &ResultForwardSwapRate,      // ResultForwardRate
    double &ExerciseValue               // 행사 Value
)
{

    long i;
    long idx = 0;
    double FSR;

    double T_Option = ((double)DayCountAtoB(PriceDate, StartDate))/365.0;
    if (T_Option < 0.0000285388)
    {
        // 옵션만기까지 최소 15분
        T_Option = 0.0000285388;
    }
    double dt, t_pay, value, d1, d2, value_atm , d1_atm, d2_atm;
    double annuity = 0.0;
    FSR = ForwardSwapRate(PriceDate, StartDate, NCpn, SwapDate, NTerm, Term, Rate, NTerm, Term, Rate, DayCountFracFlag);
    ResultForwardSwapRate = FSR;

    for (i = 0; i < NCpn; i++)
    {
        if (i == 0) dt = DayCountFrc(StartDate, SwapDate[i], DayCountFracFlag);
        else dt = DayCountFrc(SwapDate[i - 1], SwapDate[i], DayCountFracFlag);
        t_pay = ((double)DayCountAtoB(PriceDate, SwapDate[i])) / 365.0;
        annuity += dt * Calc_DiscountFactor_Pointer(Term, Rate, NTerm, t_pay, idx);
    }

    value = 0.0;
    value_atm = 0.0;
```

Forward Swap Rate 계산(StartDate to SwapDate[NCpn-1])

```
    if (VolFlag == 0)        Black Vol일 경우
    {
        d1 = (log(FSR / StrikePrice) + 0.5 * Vol * Vol * T_Option) / (Vol * sqrt(T_Option));
        d2 = d1 - Vol * sqrt(T_Option);

        d1_atm = 0.5 * Vol * sqrt(T_Option);
        d2_atm = -0.5 * Vol * sqrt(T_Option);

        if (PriceDate != StartDate)
        {
            if (FixedPayer == 0)
            {
                value = max(annuity * (FSR * CDF_N(d1) - StrikePrice * CDF_N(d2)), 0.0);
                value_atm = annuity * (FSR * CDF_N(d1_atm) - FSR * CDF_N(d2_atm));
            }
            else
            {
                value = max(annuity * (-FSR * CDF_N(-d1) + StrikePrice * CDF_N(-d2)), 0.0);
                value_atm = annuity * (-FSR * CDF_N(-d1_atm) + FSR * CDF_N(-d2_atm));
            }
        }
        else
        {
            if (FixedPayer == 0)
            {
                value = annuity * max(FSR - StrikePrice, 0.0);
                value_atm = 0.0;
            }
            else
            {
                value = annuity * max(-FSR + StrikePrice, 0.0);
                value_atm = 0.0;
            }
        }
    }
    else
    {
        d1 = (FSR - StrikePrice) / (Vol * sqrt(T_Option));        Normal Vol의 경우
        d1_atm = 0.0;
        if (PriceDate != StartDate)
        {
            if (FixedPayer == 0)
            {
                value = max(annuity * ((FSR - StrikePrice) * CDF_N(d1) + Vol * sqrt(T_Option) * (exp(-d1 * d1 / 2.0) / 2.506628274631)), 0.0);
                value_atm = annuity * Vol * sqrt(T_Option) * (exp(-d1_atm * d1_atm / 2.0) / 2.506628274631);
            }
            else
            {
                value = max(annuity * ((FSR - StrikePrice) * CDF_N(d1) + Vol * sqrt(T_Option) * (exp(-d1 * d1 / 2.0) / 2.506628274631)), 0.0);
                value_atm = annuity * Vol * sqrt(T_Option) * (exp(-d1_atm * d1_atm / 2.0) / 2.506628274631);
                value = max(0.0 , annuity * (StrikePrice - FSR) + value);
            }
        }
        else
        {
            if (FixedPayer == 0)
            {
                value = max(annuity * ((FSR - StrikePrice) * CDF_N(d1) + Vol * sqrt(T_Option) * (exp(-d1 * d1 / 2.0) / 2.506628274631)), 0.0);
                value_atm = annuity * Vol * sqrt(T_Option) * (exp(-d1_atm * d1_atm / 2.0) / 2.506628274631);
            }
            else
            {
                value = max(annuity * ((FSR - StrikePrice) * CDF_N(d1) + Vol * sqrt(T_Option) * (exp(-d1 * d1 / 2.0) / 2.506628274631)), 0.0);
                value_atm = annuity * Vol * sqrt(T_Option) * (exp(-d1_atm * d1_atm / 2.0) / 2.506628274631);
                value = max(0.0 , annuity * (StrikePrice - FSR) + value);
            }
        }
        else
        {
            if (FixedPayer == 0)
            {
                value = annuity * max(FSR - StrikePrice, 0.0);
            }
            else
            {
                value = annuity * max(-FSR + StrikePrice, 0.0);
            }
            value_atm = 0.0;
        }
    }
    ExerciseValue = (FSR - StrikePrice) * annuity;
    if (PricingOrValueFlag == 0) return NA * value;
    else return NA * (value - value_atm);
}
```