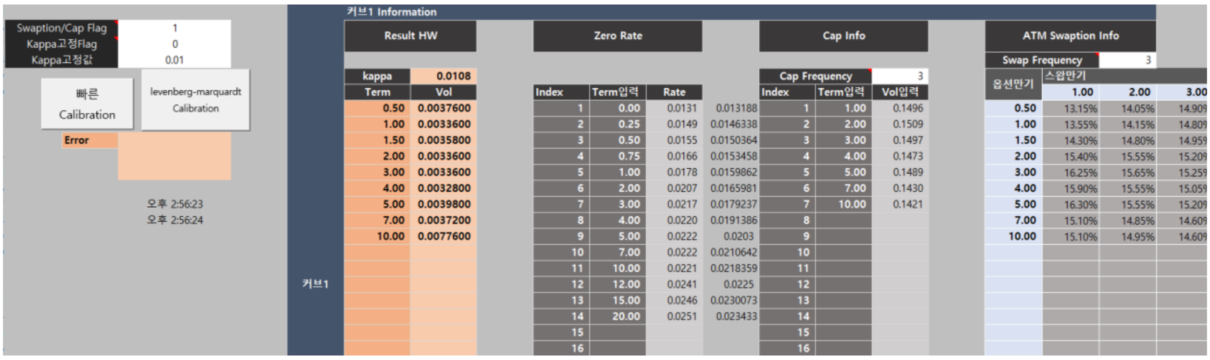
HW_Calibration 로직 설명서

빠른 Calibration, Levenberg-marquardt 알고리즘 두 가지 사용 가능



Swaption/Cap Flag	1
Kappa고정Flag	0
Kappa고정값	0.01

Result HW

kappa	0.0108
Term	Vol
0.50	0.0037600
1.00	0.0033600
1.50	0.0035800
2.00	0.0033600
3.00	0.0033600
4.00	0.0032800
5.00	0.0039800
7.00	0.0037200
10.00	0.0077600

Zero Rate

Index	Term입력	Rate	
1	0.00	0.0131	0.013188
2	0.25	0.0149	0.0146338
3	0.50	0.0155	0.0150364
4	0.75	0.0166	0.0153458
5	1.00	0.0178	0.0159862
6	2.00	0.0207	0.0165981
7	3.00	0.0217	0.0179237
8	4.00	0.0220	0.0191386
9	5.00	0.0222	0.0203
10	7.00	0.0222	0.0210642
11	10.00	0.0221	0.0218359
12	12.00	0.0241	0.0225
13	15.00	0.0246	0.0230073
14	20.00	0.0251	0.023433
15			
16			

Cap Info

Cap Frequency		3
Index	Term입력	Vol입력
1	1.00	0.1496
2	2.00	0.1509
3	3.00	0.1497
4	4.00	0.1473
5	5.00	0.1489
6	7.00	0.1430
7	10.00	0.1421
8		
9		
10		
11		
12		
13		
14		
15		
16		

ATM Swaption Info

Swap Frequency 3

옵션만기	스왑만기		
	1.00	2.00	3.00
0.50	13.15%	14.05%	14.90
1.00	13.55%	14.15%	14.80
1.50	14.30%	14.80%	14.95
2.00	15.40%	15.55%	15.20
3.00	16.25%	15.65%	15.25
4.00	15.90%	15.55%	15.05
5.00	16.30%	15.55%	15.20
7.00	15.10%	14.85%	14.60
10.00	15.10%	14.95%	14.60

1. Hull White 1Factor Swaption – Code 1, 2, 3 참고

$$SWPT = P(0,T_{opt})\left\{N\left(-\frac{lnG}{H\sqrt{V_{T_{opt}}}}+\frac{H\sqrt{V_{T_{opt}}}}{2}\right)-G\cdot N\left(-\frac{lnG}{H\sqrt{V_{T_{opt}}}}-\frac{H\sqrt{V_{T_{opt}}}}{2}\right)\right\}$$

$$G=\frac{P(0,T_N)+K\sum_{i=1}^{N}\Delta T_i P(0,T_i)}{P(0,T_{opt})}$$

$$H=\frac{P(0,T_N)B(T_{opt},T_N,\kappa)+K\sum_{i=1}^{N}\Delta T_i P(0,T_i)B(T_{opt},T_N,\kappa)}{P(0,T_N)+K\sum_{i=1}^{N}\Delta T_i P(0,T_i)}$$

$$V_{T_{opt}}=\int_0^{T_{opt}}\sigma_u^2 e^{-2\kappa(T_{opt}-u)}du$$

$$B(T_0,T_1,\kappa)=\frac{1-e^{-\kappa(t-s)}}{\kappa}$$

2. Hull White 2Factor Swaption – Code 4~10 참고

$$SWPT = P(0,T_{opt})\int_{-\infty}^{\infty}\frac{e^{-\frac{(x-\mu_x)^2}{2\sigma_x^2}}}{\sigma_x\sqrt{2\pi}}\left\{N(-h_1(x))-\sum_{i=1}^{n}\lambda_i(x)e^{\kappa_i(x)}N\left(-h_{2,i}(x)\right)\right\}dx$$

$$h_1(x)=\frac{\bar{y}(x)-\mu_y}{\sigma_y\sqrt{1-\rho_{xy}^2}}-\frac{\rho_{xy}(x-\mu_x)}{\sigma_x\sqrt{1-\rho_{xy}^2}}$$

$$h_{2,i}(x)=h_1(x)+B_2(T_0,T_i)\sigma_y\sqrt{1-\rho_{xy}^2}$$

$$\lambda_i(x)=c_i A(T_0,T_i)e^{-B_1(T_0,T_i)x}$$

$$\kappa_i(x) = -B_2(T_0,T_i)\left\{\mu_y - \frac{1}{2}(1-\rho_{xy}^2)\sigma_y^2 B_2(T_0,T_i) + \frac{\rho_{xy}\sigma_y(x-\mu_x)}{\sigma_x}\right\}$$

$$A(T_0,T_i) = \frac{P^M(0,T_i)}{P^M(0,T_0)} e^{QVT_x(T_0,T_i)+QVT_y(T_0,T_i)+QVT_{xy}(T_0,T_i)}$$

$$QVT_x(T_0,T_i) = \frac{1}{2}\left(V_{\kappa_1}(T_0,T_i) + V_{\kappa_1}(0,T_i) + V_{\kappa_1}(0,T_0)\right)$$

$$QVT_{xy}(T_0,T_i) = \frac{1}{2}\left(V_{\kappa_1,\kappa_2}(T_0,T_i) + V_{\kappa_1,\kappa_2}(0,T_i) + V_{\kappa_1,\kappa_2}(0,T_0)\right)$$

$$V_{\kappa_1}(t,T) \approx \frac{\sigma_1^2}{\kappa_1^2}\left(T - t + 2\frac{e^{-\kappa_1(T-t)}-1}{\kappa_1} - \frac{e^{-2\kappa_1(T-t)}-1}{2\kappa_1}\right)$$

$$V_{\kappa_1,\kappa_2}(t,T) \approx \frac{\sigma_1\sigma_2}{\kappa_1\kappa_2}\left(T - t + \frac{e^{-\kappa_1(T-t)}-1}{\kappa_1} + \frac{e^{-\kappa_2(T-t)}-1}{\kappa_2} - \frac{e^{-(\kappa_1+\kappa_2)(T-t)}-1}{\kappa_1+\kappa_2}\right)$$

$$M_x^T(s,t) = \int_s^t e^{-\kappa(t-u)}\left\{\sigma_1^2 B_{\kappa_1}(u,T) - \rho\sigma_1\sigma_2 B_{\kappa_2}(u,T)\right\}du$$

$$\mu_x = -M_x^T(0,T_0)$$

$$\sigma_x = \int_0^{T_0}\sigma_1^2(u)e^{-2\kappa(T_0-u)}du \approx \frac{\sigma_1^2}{2\kappa_1}(1-e^{-2\kappa_1 T_0})$$

$$\rho_{xy} = \frac{\rho\int_0^{T_0}\sigma_1(u)\sigma_2(u)e^{-(\kappa_1+\kappa_2)(T_0-u)}du}{\sigma_x\sigma_y}$$

이고, $\bar{y}(x)$는 주어진 x에 대하여

$$\sum_{i=1}^{n} c_i A(T_0,T_i)e^{-B_1(T_0,T_i)x - B_2(T_0,T_i)\bar{y}(x)} = 1$$

$$c_i = K\Delta T_i, \qquad i = 1,2,\dots,n-1, c_n = 1 + K\Delta T_n$$

3. Levenberg-marquardt

$$(\kappa^*, \sigma_t^*) = argmin_{\kappa,\sigma}\sum_i\left\{BLACKPRICE_i^{mkt} - HWPRICE(\kappa,\sigma_t)\right\}^2$$

$$P_{k+1} = P_k - (J^TJ + \mu_k I)^{-1}J^T R(p_k)$$

(간혹 $\mu_k I$ *대신에* $\mu_k(J^TJ)$*를 사용하기도 함*)

여기서 $J = \begin{bmatrix} \frac{\delta r_1(p)}{\delta p_1} & \cdots & \frac{\delta r_1(p)}{\delta p_m} \\ \vdots & \ddots & \vdots \\ \frac{\delta r_n(p)}{\delta p_1} & \cdots & \frac{\delta r_n(p)}{\delta p_m} \end{bmatrix}$, $R(p) = \begin{bmatrix} r_1(p) \\ \vdots \\ r_n(p) \end{bmatrix}$

4. 빠른 Calibration 1F

1-1 빠른 Calibration 방법론

κ는 0.002부터 0.1까지 0.002간격으로,

σ_t는 0.001부터 0.04까지 0.001간격으로 넣고 Swaption 및 Cap Pricing을 한다.

$\mathrm{for}(\kappa = 0.002 \;\; \mathrm{to} \;\; 0.1; \;\; d\kappa = 0.002)$

$\quad \mathrm{for}(\sigma = 0.001 \;\; \mathrm{to} \;\; 0.04; \;\; d\sigma = 0.001)$

$\quad\quad \mathrm{Error}(\kappa, \sigma) = \mathrm{P}(\kappa, \sigma) - \mathrm{P}(\mathrm{black})$

$\mathrm{Find \;\; Min \;\; Error \;\; Point}(\kappa^*, \sigma^*)$

찾아낸 κ, σ근방에서 위의 로직을 한 번 더 실행함

$\mathrm{for}(\kappa = \hat{\kappa} - 0.001 \;\; \mathrm{to} \;\; \hat{\kappa} + 0.001; \;\; d\hat{\kappa} = 0.0002)$

$\quad \mathrm{for}(\sigma = \hat{\sigma} - 0.001 \;\; \mathrm{to} \;\; \hat{\sigma} + 0.001; \;\; d\hat{\sigma} = 0.0001)$

$\quad\quad \mathrm{Error}(\kappa, \sigma) = \mathrm{P}(\kappa, \sigma) - \mathrm{P}(\mathrm{black})$

$\mathrm{Find \;\; Min \;\; Error \;\; Point}(\kappa, \sigma)$

1-2 Calibration 예시

Example) 다음과 같이 Swaption Vol이 주어진다고 가정하자.

	Swapmat= 1	Swapmat= 2	Swapmat= 3
Optmat= 0.5	10%	12%	14%
Optmat= 1	11%	13%	15%
Optmat= 1.5	12%	14%	16%

Calibration은 다음과 같이 실행된다.

$\mathrm{for}(\mathrm{optmat} = 0.5 \ \ \mathrm{to} \ \ 1.5)$

$\mathrm{for}(\kappa = 0.002 \ \ \mathrm{to} \ \ 0.1; \ \ d\kappa = 0.002)$

$\mathrm{for}(\sigma = 0.001 \ \ \mathrm{to} \ \ 0.04; \ \ d\sigma = 0.001)$

{

$\mathrm{Error}_1(\kappa, \sigma) = \mathrm{P}(\kappa, \sigma) - \mathrm{P}(\mathrm{black}, \mathrm{Vol}_1)$

$\mathrm{Error}_2(\kappa, \sigma) = \mathrm{P}(\kappa, \sigma) - \mathrm{P}(\mathrm{black}, \mathrm{Vol}_2)$

$\mathrm{Error}_3(\kappa, \sigma) = \mathrm{P}(\kappa, \sigma) - \mathrm{P}(\mathrm{black}, \mathrm{Vol}_3)$

$Error = Error_1 + Error_2 + Error_3$

}

$\mathrm{Find \ \ Min \ \ Error \ \ Point}(\kappa, \sigma_{\mathrm{optmat}})$

찾아낸 κ, σ근방에서 위의 로직을 한 번 더 실행함

Code 1. $\boldsymbol{B(T_0, T_1, \kappa) = \frac{1 - e^{-\kappa(t-s)}}{\kappa}}$

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

Code 2. $\boldsymbol{V_{T_{opt}} = \int_0^{T_{opt}} \sigma_u^2 e^{-2\kappa(T_{opt}-u)} du}$

$$\boldsymbol{= e^{-2\kappa T_{opt}} \cdot \int_0^{T_{opt}} \sigma_u^2 e^{2\kappa u} du = e^{-2\kappa T_{opt}} \times Integ(T_{opt}, 1, 2\kappa, \{\sigma\})}$$

```
// 적분 계산 공통함수
// I(t) = Int_0^t sigma(s)^2 A exp(ks) ds
double Integ(
    double t,
    double A,
    double kappa,
    double* tVol,
    double* Vol,
    long nVol
)
{
    long i;
    long NodeNum = 10;
    long Point = 0;
    double ds = t / (double)NodeNum;
    double s;
    double value = 0.0;
    double sigma;
    if (nVol == 1) return A * Vol[0] * Vol[0] / (kappa) * (exp(kappa * t) - 1.0);
```

```
    for (i = 0; i < NodeNum; i++)
    {
        s = (double)(i + 1) * ds;
        sigma = Interpolate_Linear_Point(tVol, Vol, nVol, s, Point);
        value += sigma * sigma * A * exp(kappa * s) * ds;
    }
    return value;
}
```

Code 3. Swaption Pricing

```
// 1-factor 모형의 Fixed Payer Swaption 가격 빠른 계산
double HW_Swaption(
    double NA,              // 액면금액
    double kappa,           // 회귀속도
    double* tVol,           // 변동성 구간 종점
    double* Vol,            // 구간 변동성
    long nVol,              // 변동성 구간 개수
    double* t,              // 할인채 만기
    double* r,              // 할인채 가격
    long nr,                // 할인채 개수
    double StrikeRate,      // 고정금리(지급부분)
    long MaturityDate,      // 옵션 만기일까지 일수
    long* Dates,            // 지급일: 계산일로부터 각 쿠폰지급일까지의 일수
    long nDates,            // 지급 회수(계산일 이후 남은 회수
    double* PT              // P[0] = P(0, T0pt), P[N] = P(0, TSwp), Shape=nDates + 1
)
{
    long i;
    double T0, PrevT, T, deltaT;

    double VT0, G, H;
    double d1, d2;
    double value;

    if (kappa < -0.002) kappa = -0.002;

    for (i = 0; i < nVol; i++)
    {
        if (Vol[i] < 0.0) Vol[i] = -Vol[i];
        if (Vol[i] < Tiny_Value) Vol[i] = Tiny_Value;
    }

    T0 = (double)MaturityDate / 365.0;
    VT0 = exp(-2.0 * kappa * T0) * Integ(T0, 1.0, 2.0 * kappa, tVol, Vol, nVol);
    G = PT[nDates];
    PrevT = T0;
    for (i = 0; i < nDates; i++)
    {
        T = (double)Dates[i] / 365.0;
        deltaT = T - PrevT;
        G += StrikeRate * deltaT * PT[i + 1];
        PrevT = T;
    }
    G /= PT[0];

    H = PT[nDates] * B(T0, (double)Dates[nDates - 1] / 365.0, kappa);
    PrevT = T0;
```

```
    for (i = 0; i < nDates; i++) {
        T = (double)Dates[i] / 365.0;
        deltaT = T - PrevT;
        H += StrikeRate * deltaT * PT[i + 1] * B(T0, T, kappa);
        PrevT = T;
    }
    H /= G * PT[0];

    d1 = -log(G) / (H * sqrt(VT0)) + 0.5 * H * sqrt(VT0);
    d2 = -log(G) / (H * sqrt(VT0)) - 0.5 * H * sqrt(VT0);

    value = PT[0] * (CDF_N(d1) - G * CDF_N(d2));
    return NA * value;
}
```

Code 4. $M_x^T(s,t) = \int_s^t e^{-\kappa(t-u)}\{\sigma_1^2 B_{\kappa_1}(u,T) - \rho\sigma_1\sigma_2 B_{\kappa_2}(u,T)\}\,du$

```
double mu_x(
     double kappa1, double kappa2, long nHW1,
     double *HWTerm1, double* HWVol1, long nHW2,
     double* HWTerm2, double* HWVol2, double rho,
     double T, double t1, double t2
)
{
    long i;
    long NodeNum = 10;
    long Point = 0;
    long Point2 = 0;
    double du = (t2-t1) / (double)NodeNum;
    double u = t1;
    double value = 0.0;
    double sigma;
    double v1, v2;
    for (i = 0; i < NodeNum; i++)
    {
        v1 = Interpolate_Linear_Point(HWTerm1, HWVol1, nHW1, u, Point);
        v2 = Interpolate_Linear_Point(HWTerm2, HWVol2, nHW2, u, Point2);
        value += exp(-kappa1 * (t2 - u)) * (v1 * v1 * B(u, T, kappa1) - rho * v1 * v2 *
B(u, T, kappa2)) * du;
        u += du;
    }
    return value;
}
```

Code 5. $V_{\kappa_1}(t,T) \approx \frac{\sigma_1^2}{\kappa_1^2}\left(T - t + 2\frac{e^{-\kappa_1(T-t)}-1}{\kappa_1} - \frac{e^{-2\kappa_1(T-t)}-1}{2\kappa_1}\right)$

```
double V(
    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));
```

```
}
```

Code 6. $QVT_x(T_0,T_i)=\frac{1}{2}\left(V_{\kappa_1}(T_0,T_i)+V_{\kappa_1}(0,T_i)+V_{\kappa_1}(0,T_0)\right)$

```
double QV(
    double t, double T, double kappa,
    long NHWVol, double* HWVolTerm, double* HWVol
)
{
    long i;
    double vol;
    double RHS = 0.0;

    if (NHWVol == 1 || kappa > 0.1)
    {
        vol = Interpolate_Linear(HWVolTerm, HWVol, NHWVol, t);
        RHS = 0.5 *(V(kappa,kappa,t,T,vol,vol)-V(kappa,kappa,0,T,vol,vol)+V(kappa,kappa,0,
t,vol,vol));
    }
    else
    {
        RHS = 0.0;
        long NInteg = 10;
        double u = t;
        double du = (T - t) / ((double)NInteg);
        double Bst, BsT;
        for (i = 0; i < NInteg; i++)
        {
            vol = Interpolate_Linear(HWVolTerm, HWVol, NHWVol, u);
            Bst = B(u, t, kappa);
            BsT = B(u, T, kappa);
            RHS += 0.5 * vol * vol * (Bst * Bst - BsT * BsT) * du;
            u = u + du;
        }
    }
    return RHS;
}
```

Code 7. $QVT_{xy}(T_0,T_i)=\frac{1}{2}\left(V_{\kappa_1,\kappa_2}(T_0,T_i)+V_{\kappa_1,\kappa_2}(0,T_i)+V_{\kappa_1,\kappa_2}(0,T_0)\right)$

```
double CQV(
    double t, double T, double kappa,
    long NHWVol, double* HWVolTerm, double* HWVol,
    double kappa2, double* HWVolTerm2, double* HWVol2,
    double rho
)
{
    long i;
    double Bst, BsT, vol, vol2;
    double RHS = 0.0;
    double s, ds;

    long NInteg = 10.0;
```

```
    double u = t;
    double du = (T - t) / ((double)NInteg);
    RHS = 0.0;
    if (NHWVol > 1)
    {
        vol = 0.5*Interpolate_Linear(HWVolTerm,HWVol,NHWVol,t)+0.5*
Interpolate_Linear(HWVolTerm, HWVol, NHWVol, T);
        vol2 = 0.5*Interpolate_Linear(HWVolTerm,HWVol2,NHWVol,t)+0.5*
Interpolate_Linear(HWVolTerm, HWVol2, NHWVol, T);
    }
    else
    {
        vol = Interpolate_Linear(HWVolTerm, HWVol, NHWVol, t);
        vol2 = Interpolate_Linear(HWVolTerm, HWVol2, NHWVol, t);
    }

    RHS = 2.0 * rho * 0.5 * (V(kappa, kappa2, t, T, vol, vol2) - V(kappa, kappa2, 0, T,
vol, vol2) + V(kappa, kappa2, 0, t, vol, vol2));
    return RHS;
}
```

Code 8. $A(T_0, T_i) = \frac{P^M(0,T_i)}{P^M(0,T_0)} e^{QVT_x(T_0,T_i)+QVT_y(T_0,T_i)+QVT_{xy}(T_0,T_i)}$

```
// Swaption2F 함수에서 사용
double A(
    double t, double T, double kappa1,
    double kappa2, double* tVol,double* Vol1,
    double* Vol2, double* Vol12,long nVol,
    double rho, double DF_t, double DF_T
)
{
return exp(QV(t,T,kappa1,nVol,tVol,Vol1)+QV(t,T,kappa2,nVol,tVol,Vol2)+CQV(t,T,kappa1,nVol,
tVol,Vol1,kappa2,tVol,Vol2,rho));
}
```

Code 9. $\int_{-\infty}^{\infty} \frac{e^{-\frac{(x-\mu_x)^2}{2\sigma_x^2}}}{\sigma_x\sqrt{2\pi}} f(x)dx = \sum_{i=1}^{n} \frac{e^{-\frac{(x_i-\mu_x)^2}{2\sigma_x^2}}}{\sigma_x\sqrt{2\pi}} f(x_i) \times w_i$

```
// Gauss Hermite Normal 적분 계산에 사용될 x지점과 weight를 계산
// integral_-inf to inf 1.0/(sigma*sqrt(2.0PI)) * exp(-((x-mu)/sigma)^2) f(x) dx = sum(w *
f(x))
void gauss_hermite_normal(double* x, double* w, double mu, double sigma, long n)
{
        long i;
        double sqrt2 = 1.4142135623730951;//sqrt(2.0);
        double sqrtPI = 1.772453809055159;//sqrt(PI);
        gauss_hermite(x, w, n);

        for (i = 0; i < n; i++)
        {
                x[i] = (x[i] * sqrt2) * sigma + mu;
                w[i] = (w[i] / sqrtPI);
        }
```

```
}
```

Code 10. $SWPT = P(0, T_{opt}) \int_{-\infty}^{\infty} \frac{e^{-\frac{(x-\mu_x)^2}{2\sigma_x^2}}}{\sigma_x\sqrt{2\pi}} \left\{ N(-h_1(x)) - \sum_{i=1}^{n} \lambda_i(x) e^{\kappa_i(x)} N\left(-h_{2,i}(x)\right) \right\} dx$

```
// 2-factor 모형의 Fixed Payer Swaption 가격 계산
double _stdcall Swaption2F(
    double NA, double kappa1, double kappa2,
    double* tVol, double* Vol1, double* Vol2,
    double* Vol12, long nVol, double rho,
    double StrikeRate, long MaturityDate, long* Dates,
    double* termdates, double* termC, long nDates,
    double* PT, double P0_at_OptMaturity,
    long nQuad,                 //Gauss Normal Quadrature 개수
    double* x,                  //Gauss Normal Quadrature의 x값
    double* w                   //Gauss Normal Quadrature의 y값 비율
)
{
    long i, j;
    double sum, value = 0.0;
    double T;
    double h1, h2, kappa_i, y;
    double m_x, sigma_x, m_y, sigma_y, rho_xy;

    if (kappa1 < Tiny_Value) kappa1 = Tiny_Value;
    if (kappa2 < Tiny_Value) kappa2 = Tiny_Value;

    T = (double)MaturityDate / 365.0;
    double exp_minus_kappa1_T = exp(-kappa1 * T);
    double exp_minus2_kappa1_T = exp(-2.0 * kappa1 * T);
    double exp_minus_kappa2_T = exp(-kappa2 * T);
    double exp_minus2_kappa2_T = exp(-2.0 * kappa2 * T);

    m_x = -mu_x(kappa1, kappa2, nVol, tVol, Vol1, nVol, tVol, Vol2, rho, T, 0, T);
    m_y = -mu_x(kappa2, kappa1, nVol, tVol, Vol2, nVol, tVol, Vol1, rho, T, 0, T);
    double v1, v2;
    v1 = Interpolate_Linear(tVol, Vol1, nVol, T);
    v2 = Interpolate_Linear(tVol, Vol2, nVol, T);
    sigma_x = sqrt(v1 * v1 * (1.0 - exp(-2.0 * kappa1 * T)) / (2.0 * kappa1));
    sigma_y = sqrt(v2 * v2 * (1.0 - exp(-2.0 * kappa2 * T)) / (2.0 * kappa2));
    rho_xy = rho * exp(-(kappa1 + kappa2) * T) * Integ(T, 1.0, kappa1 + kappa2, tVol, Vol12, nVol) /
(sigma_x * sigma_y);
    if (sigma_x < 0.0) sigma_x = -sigma_x;
    if (sigma_x < Tiny_Value) sigma_x = Tiny_Value;
    if (sigma_y < 0.0) sigma_y = -sigma_y;
    if (sigma_y < Tiny_Value) sigma_y = Tiny_Value;

    rho_xy = max(-0.9999,min(0.9999,rho_xy));

    gauss_hermite_normal(x, w, m_x, sigma_x, nQuad);
    for (i = 0; i < nQuad; i++) {
        for (j = 0; j < nDates; j++)
        {
            if (j == 0)
            {
                termC[j] = StrikeRate * (termdates[j] - T) * A(T, termdates[j], kappa1, kappa2, tVol,
Vol1, Vol2, Vol12, nVol, rho, P0_at_OptMaturity, PT[j]) * exp(-B(T, termdates[j], kappa1) * x[i]);
            }
            else if (j == nDates - 1)
            {
                termC[j] = (1.0 + StrikeRate * (termdates[j] - termdates[j - 1])) * A(T, termdates[j],
kappa1, kappa2, tVol, Vol1, Vol2, Vol12, nVol, rho, P0_at_OptMaturity, PT[j]) * exp(-B(T,
termdates[j], kappa1) * x[i]);
            }
```

```
            else
            {
                termC[j] = StrikeRate * (termdates[j] - termdates[j - 1]) * A(T, termdates[j], kappa1, kappa2, tVol, Vol1, Vol2, Vol12, nVol, rho, P0_at_OptMaturity, PT[j]) * exp(-B(T, termdates[j], kappa1) * x[i]);
            }
        }

        y = Find_Sol2(kappa2, termC, T, termdates, nDates);

        h1 = ((y - m_y) / sigma_y - rho_xy * (x[i] - m_x) / sigma_x) / sqrt(1.0 - rho_xy * rho_xy);

        sum = CDF_N(-h1);
        for (j = 0; j < nDates; j++)
        {
            h2 = h1 + B(T, termdates[j], kappa2) * sigma_y * sqrt(1.0 - rho_xy * rho_xy);
            kappa_i = -B(T, termdates[j], kappa2) * (m_y - 0.5 * (1.0 - rho_xy * rho_xy) * sigma_y * sigma_y * B(T, termdates[j], kappa2) + rho_xy * sigma_y * (x[i] - m_x) / sigma_x);
            sum -= termC[j] * exp(kappa_i) * CDF_N(-h2);
        }

        value += w[i] * sum;
    }

    return NA * value * P0_at_OptMaturity;
}
```

Code 11.