

## [Help](#)

```
#include "
href../../mod/hullwhite2d/hullwhite2d_std/hullwhite2d_std_h_src.pdfhullwhit
#include "
href../../mod/hullwhite2d/hullwhite2d_std/hullwhite2d_includes_h_src.pdfhull
#include "pnl/pnl_cdf.h"

//The "#else" part of the code will be freely available after the (year of creat
#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2009+2)
int CALC(CF_FLOORHW2D)(void *Opt, void *Mod, PricingMethod *Met)
{
    return AVAILABLE_IN_FULL_PREMIA;
}
static int CHK_OPT(CF_FLOORHW2D)(void *Opt, void *Mod)
{
    return NONACTIVE;
}
#else

// Volatility of an european option on a ZC bond P(T,S)
static double cf_ZB0volatility2d(double a, double sigma1, double b, double sigma2,
{
    double sigma_p;
    //double U, V, B_TS;
    double exp_atT, exp_btT, exp_aTS, exp_bTS;
    double sigma3, eta, rhoG2;

    sigma3 = sqrt(sigma1 * sigma1 + sigma2 * sigma2 / ((b - a) * (b - a)) + 2 * rh
    eta = sigma2 / (a - b);
    rhoG2 = (sigma1 * rho - eta) / sigma3 ;

    exp_atT = exp(-a * (T - t));
    exp_btT = exp(-b * (T - t));

    exp_aTS = exp(-a * (S - T));
    exp_bTS = exp(-b * (S - T));

    /* B_TS = (1 - exp_aTS) / a; */
    /* U = (exp_aTS - 1) * exp_atT/(a*(a-b)); //(1/exp_aS - 1/exp_aT)/(a*(a-b)); */
    /* V = (exp_bTS - 1) * exp_btT/(b*(a-b)); // (1/exp_bS - 1/exp_bT)/(b*(a-b)); */
```

```

sigma_p = sigma3 * sigma3 * (1 - exp_aTS) * (1 - exp_aTS) * (1 - exp_atT * ex
sigma_p += eta * eta * (1 - exp_bTS) * (1 - exp_bTS) * (1 - exp_btT * exp_btT)
sigma_p += 2 * rhoG2 * sigma3 * eta * (1 - exp_aTS) * (1 - exp_bTS) * (1 - exp
sigma_p = sqrt(sigma_p);

return sigma_p;
}

```

```

static double cf_zbcall2d(ZCMarketData *ZCMarket, double t, double r, double u,
{
    double PtS, PtT;
    double h, sigma_p;
    double price;

    sigma_p = cf_ZB0volatility2d(a, sigma1, b, sigma2, rho, t, T, S);

    PtT = cf_hw2d_zcb(ZCMarket, a, sigma1, b, sigma2, rho, t, r, u, T);

    PtS = cf_hw2d_zcb(ZCMarket, a, sigma1, b, sigma2, rho, t, r, u, S);

    h = log(PtS / (PtT * X)) / sigma_p + 0.5 * sigma_p ;

    price = PtS * cdf_nor(h) - X * PtT * cdf_nor(h - sigma_p);

    return price;
}

```

```

static int cf_floor2d(int flat_flag, double r_t, char *curve, double u_t, double
double Nominal, double K, double periodicity, double first
{

    double sum, tim, tip, strike_put;
    int i, nb_payement;
    ZCMarketData ZCMarket;

```

```

/* Flag to decide to read or not ZC bond datas in "initialyields.dat" */
/* If P(0,T) not read then P(0,T)=exp(-r0*T) */
if (flat_flag == 0)
{
    ZCMarket.FlatOrMarket = 0;
    ZCMarket.Rate = r_t;
}

else
{
    ZCMarket.FlatOrMarket = 1;
    ZCMarket.filename = curve;
    ReadMarketData(&ZCMarket);

    if (contract_maturity > GET(ZCMarket.tm, ZCMarket.Nvalue - 1))
    {
        printf("\ nError : time bigger than the last time value entered in ini
        exit(EXIT_FAILURE);
    }
}

strike_put = 1. / (1 + periodicity * K);
nb_payment = (int)((contract_maturity - first_payment) / periodicity);

/*Cap=Portfolio of zero-bond Put options*/
sum = 0.;
for (i = 0; i < nb_payment; i++)
{
    tim    = first_payment + (double)i * periodicity;
    tip    = tim + periodicity;
    sum    += cf_zbcall2d(&ZCMarket, 0, r_t, u_t, a, sigma1, b, sigma2, rho, tip
}

sum = Nominal * (1. + K * periodicity) * sum;

/*Price*/
*price = sum;

DeleteZCMarketData(&ZCMarket);

```

```

    return OK;
}

int CALC(CF_FLOORHW2D)(void *Opt, void *Mod, PricingMethod *Met)
{
    TYPEOPT *ptOpt = (TYPEOPT *)Opt;
    TYPEMOD *ptMod = (TYPEMOD *)Mod;

    return cf_floor2d(ptMod->flat_flag.Val.V_INT,
                      MOD(GetYield)(ptMod),
                      MOD(GetCurve)(ptMod),
                      ptMod->InitialYieldsu.Val.V_PDOUBLE,
                      ptMod->aR.Val.V_DOUBLE,
                      ptMod->SigmaR.Val.V_PDOUBLE,
                      ptMod->bu.Val.V_DOUBLE,
                      ptMod->Sigmau.Val.V_PDOUBLE,
                      ptMod->Rho.Val.V_PDOUBLE,
                      ptOpt->Nominal.Val.V_PDOUBLE,
                      ptOpt->FixedRate.Val.V_PDOUBLE,
                      ptOpt->ResetPeriod.Val.V_DATE,
                      ptOpt->FirstResetDate.Val.V_DATE - ptMod->T.Val.V_DATE,
                      ptOpt->BMaturity.Val.V_DATE - ptMod->T.Val.V_DATE,
                      &(Met->Res[0].Val.V_DOUBLE));
}

static int CHK_OPT(CF_FLOORHW2D)(void *Opt, void *Mod)
{
    return strcmp(((Option *)Opt)->Name, "Floor");
}

#endif //PremiaCurrentVersion

static int MET(Init)(PricingMethod *Met, Option *Opt)
{
    if (Met->init == 0)
    {
        Met->init = 1;
    }

    return OK;
}

```

```

PricingMethod MET(CF_FLOORHW2D) =
{
    "CF_FloorHW2D",
    {{ " ", PREMIA_NULLTYPE, {0}, FORBID}},
    CALC(CF_FLOORHW2D),
    {{ "Price", DOUBLE, {100}, FORBID} /*, {"Delta", DOUBLE, {100}, FORBID} */ , { " ", PR
    CHK_OPT(CF_FLOORHW2D),
    CHK_ok,
    MET(Init)
} ;

```