

## [Help](#)

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

//The "#else" part of the code will be freely available after the (year of creat
#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2009+2)
int CALC(CF_ZCBONDHW2D)(void *Opt, void *Mod, PricingMethod *Met)
{
    return AVAILABLE_IN_FULL_PREMIA;
}
static int CHK_OPT(CF_ZCBONDHW2D)(void *Opt, void *Mod)
{
    return NONACTIVE;
}
#else

void ZCPrice_Coefficient(ZCMarketData *ZCMarket, double a, double sigma1, double
{
    double B_0t, B_0T, C_0t, C_0T;
    double f0_t, P0_t, P0_T, P0_t_plus, P0_t_minus;
    double exp_at, exp_aT, exp_bt, exp_bT;
    double gamma1, gamma2, gamma3, gamma4, gamma5, gamma6;
    double eta;
    double epsilon = 1e-3;

    /*Computation pure discount*/
    P0_t = BondPrice(t, ZCMarket);
    P0_T = BondPrice(T, ZCMarket);

    /*Computation of Forward rate*/
    P0_t_plus = BondPrice(t * (1. + epsilon), ZCMarket);
    P0_t_minus = BondPrice(t * (1. - epsilon), ZCMarket);
    f0_t = -(log(P0_t_plus) - log(P0_t_minus)) / (2.*t * epsilon);

    /*A,B,C coefficient*/
    exp_at = exp(a * t);
    exp_aT = exp(a * T);
    exp_bt = exp(b * t);
```

```

exp_bT = exp(b * T);

(*B_tT) = (1 - exp_at / exp_aT) / a;
B_0T = (1 - 1. / exp_aT) / a;
B_0t = (1 - 1. / exp_at) / a;

(*C_tT) = exp_at / (exp_aT * a * (a - b)) - exp_bt / (exp_bT * b * (a - b)) +
C_0T = 1. / (exp_aT * a * (a - b)) - 1. / (exp_bT * b * (a - b)) + 1. / (a * b)
C_0t = 1. / (exp_at * a * (a - b)) - 1. / (exp_bt * b * (a - b)) + 1. / (a * b)

gamma1 = (exp_at * exp_bt - 1) / (exp_aT * exp_bT * (a - b) * (a + b)) - (SQR(
gamma2 = (gamma1 + (*C_tT) - C_0T + 0.5 * (*B_tT) * (*B_tT) - 0.5 * B_0T * B_0
gamma3 = -(1 / (exp_at * exp_bt) - 1) / ((a - b) * (a + b)) + (1 / (SQR(exp_at
gamma4 = (gamma3 - C_0t - 0.5 * SQR(B_0t) + t / a + (1 / exp_at - 1) / SQR(a))
gamma5 = (0.5 * SQR(*C_tT) - 0.5 * SQR(C_0T) + gamma2) / b;
gamma6 = (gamma4 - 0.5 * SQR(C_0t)) / b;

eta = SQR(sigma1) * (1 - 1 / SQR(exp_at)) * SQR(*B_tT) / (4 * a) - rho * sigma
eta = eta - 0.5 * SQR(sigma2) * (SQR(C_0t) * (*B_tT) + gamma6 - gamma5);

(*A_tT) = (P0_T / P0_t) * exp((*B_tT) * f0_t - eta);

}

// Price of a ZC using the three coefficient A(t,T), B(t,T) and C(tT). H&W is a
double ZCPrice_Using_Coefficient(double r_t, double u_t, double A_tT, double B_t
{
    return A_tT * exp(-B_tT * r_t - C_tT * u_t);
}

// Price at date t of a ZC maturing at T, knowing that r(t)=r_t and u(t)=u_t.
double cf_hw2d_zcb(ZCMarketData *ZCMarket, double a, double sigma1, double b, do
{
    if (t == 0)
    {
        return BondPrice(T, ZCMarket);
    }
    else
    {
        double price;

```

```

        double A_tT, B_tT, C_tT;
        A_tT = 0;
        B_tT = 0;
        C_tT = 0;

        ZCPrice_Coefficient(ZCMarket, a, sigma1, b, sigma2, rho, t, T, &A_tT, &B_tT, &C_tT);
        price = ZCPrice_Using_Coefficient(r_t, u_t, A_tT, B_tT, C_tT);

        return price;
    }
}

static int cf_zcbond2d(int flat_flag, double r_t, char *curve, double u_t, double T)
{
    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 (T > GET(ZCMarket.tm, ZCMarket.Nvalue - 1))
        {
            printf("\nError : time bigger than the last time value entered in initialyields.dat\n");
            exit(EXIT_FAILURE);
        }
    }

    //Price of an option on a ZC
    *price = cf_hw2d_zcb(&ZCMarket, a, sigma1, b, sigma2, rho, 0, r_t, u_t, T);

    DeleteZCMarketData(&ZCMarket);
}

```

```

    return OK;
}

int CALC(CF_ZCBONDHW2D)(void *Opt, void *Mod, PricingMethod *Met)
{
    TYPEOPT *ptOpt = (TYPEOPT *)Opt;
    TYPEMOD *ptMod = (TYPEMOD *)Mod;

    return cf_zcbond2d(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->BMaturity.Val.V_DATE - ptMod->T.Val.V_DATE,
                        &(Met->Res[0].Val.V_DOUBLE));
}

static int CHK_OPT(CF_ZCBONDHW2D)(void *Opt, void *Mod)
{
    return strcmp(((Option *)Opt)->Name, "ZeroCouponBond");
}

#endif //PremiaCurrentVersion

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

    return OK;
}

PricingMethod MET(CF_ZCBONDHW2D) =
{

```

```

"CF_ZCBondHW2d",
{{" ", PREMIA_NULLTYPE, {0}, FORBID}},
CALC(CF_ZCBONDHW2D),
{{"Price", DOUBLE, {100}, FORBID}/*,{"Delta",DOUBLE,{100},FORBID} */, {" ", PR
CHK_OPT(CF_ZCBONDHW2D),
CHK_ok,
MET(Init)
} ;

```