

[Help](#)

```
#include "
href../../mod/hullwhite2d/hullwhite2d_std/hullwhite2d_std_h_src.pdfhullwhit
#include "pnl/pnl_vector.h"
#include "pnl/pnl_matrix.h"
#include "
href../../common/math/InterestRateModelTree/TreeHW2D/TreeHW2D_h_src.pdfmath/I
#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(TR_ZCBONDHW2D)(void *Opt, void *Mod, PricingMethod *Met)
{
    return AVAILABLE_IN_FULL_PREMIA;
}
static int CHK_OPT(TR_ZCBONDHW2D)(void *Opt, void *Mod)
{
    return NONACTIVE;
}
#else

/// TreeHW2D      : structure that contains components of the tree (see TreeHW2D.
/// ModelHW2D     : structure that contains the parameters of the Hull&White one
/// ZCMarketData : structure that contains the Zero Coupon Bond prices of the ma

/// Computation of the payoff at the final time of the tree (ie the ZCBond matur
static void ZCBond_InitialPayoffHW2D(TreeHW2D *Meth, PnlMat *OptionPriceMat2)
{
    int jminprev, jmaxprev, kminprev, kmaxprev; // jmin[i], jmax [i]

    jminprev = pnl_vect_int_get(Meth->yIndexMin, Meth->Ngrid); // jmin(Ngrid)
    jmaxprev = pnl_vect_int_get(Meth->yIndexMax, Meth->Ngrid); // jmax(Ngrid)
    kminprev = pnl_vect_int_get(Meth->uIndexMin, Meth->Ngrid); // kmin(Ngrid)
    kmaxprev = pnl_vect_int_get(Meth->uIndexMax, Meth->Ngrid); // kmax(Ngrid)

    pnl_mat_resize(OptionPriceMat2, jmaxprev - jminprev + 1, kmaxprev - kminprev + 1);

    pnl_mat_set_double(OptionPriceMat2, 1.0);
}
```

```

/// Prix of a ZC using a trinomial tree : P(s,T /r(s)=r, u(s)=u)
static double tr_hw2d_zcbond(TreeHW2D *Meth, ModelHW2D *ModelParam, ZCMarketData
{
    double OptionPrice;

    PnlMat *OptionPriceMat1; // Matrix of prices of the option at i
    PnlMat *OptionPriceMat2; // Matrix of prices of the option at i+1
    OptionPriceMat1 = pnl_mat_create(1, 1);
    OptionPriceMat2 = pnl_mat_create(1, 1);

    ///*****Parameters of the processes r, u and y *****
    /* a = (ModelParam->rMeanReversion); */
    /* sigma1 = (ModelParam->rVolatility); */
    /* b = (ModelParam->uMeanReversion); */
    /* sigma2 = (ModelParam->uVolatility); */
    /* rho = (ModelParam->correlation); */

    //sigma3 = sqrt(sigma1*sigma1 + sigma2*sigma2/((b-a)*(b-a)) + 2*rho*sigma1*sig
    //rho_y_u = (rho * sigma1 + sigma2/(b-a)) / sigma3;

    ///***** Computation of the vector of payoff at the maturity of t
    ZCBond_InitialPayoffHW2D(Meth, OptionPriceMat2);

    ///***** Computation of the vector of payoff at the maturity of t
    BackwardIterationHW2D(Meth, ModelParam, ZCMarket, OptionPriceMat1, OptionPrice

    ///***** Price of the option at time 0 *****///
    OptionPrice = MGET(OptionPriceMat2, 0, 0);

    pnl_mat_free(& OptionPriceMat1);
    pnl_mat_free(& OptionPriceMat2);

    return OptionPrice;
}

static int tr_zcbond2d(int flat_flag, double r0, char *curve, double u0, double
{

```

```

TreeHW2D Tr;
ModelHW2D ModelParams;
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 = r0;
}

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 ini
        exit(EXIT_FAILURE);
    }
}

ModelParams.rMeanReversion = a;
ModelParams.rVolatility     = sigma1;
ModelParams.uMeanReversion = b;
ModelParams.uVolatility     = sigma2;
ModelParams.correlation     = rho;

if (a - b == 0)
{
    printf("\ nError : \ "Speed of Mean Reversion Interest Rate\ " and \ "Spee
    exit(EXIT_FAILURE);
}

// Construction of the Time Grid
SetTimegridHW2D(&Tr, N_steps, T);

// Construction of the tree, calibrated to the initial yield curve

```

```

SetTreeHW2D(&Tr, &ModelParams, &ZCMarket);

//Price of Zero Coupon Bond
*price = tr_hw2d_zcbond(&Tr, &ModelParams, &ZCMarket, T, N_steps, r0, u0);

DeleteTreeHW2D(&Tr);
DeleteZCMarketData(&ZCMarket);

return OK;
}

///***** PREMIA FUNCTIONS *****/

int CALC(TR_ZCBONDHW2D)(void *Opt, void *Mod, PricingMethod *Met)
{
    TYPEOPT *ptOpt = (TYPEOPT *)Opt;
    TYPEMOD *ptMod = (TYPEMOD *)Mod;

    return tr_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->Par[0].Val.V_LONG,
        &(Met->Res[0].Val.V_DOUBLE));
}

static int CHK_OPT(TR_ZCBONDHW2D)(void *Opt, void *Mod)
{
    if ((strcmp(((Option *)Opt)->Name, "ZeroCouponBond") == 0))
        return OK;
    else

```

```

        return WRONG;
    }
#endif //PremiaCurrentVersion

static int MET(Init)(PricingMethod *Met, Option *Opt)
{
    if (Met->init == 0)
    {
        Met->init = 1;
        Met->Par[0].Val.V_LONG = 50;
    }
    return OK;
}

PricingMethod MET(TR_ZCBONDHW2D) =
{
    "TR_ZCBondHW2D",
    { {"TimeStepNumber", LONG, {100}, ALLOW},
      {" ", PREMIA_NULLTYPE, {0}, FORBID}
    },
    CALC(TR_ZCBONDHW2D),
    {{"Price", DOUBLE, {100}, FORBID}/*,{"Delta",DOUBLE,{100},FORBID} */ , {" ", PR
    CHK_OPT(TR_ZCBONDHW2D),
    CHK_ok,
    MET(Init)
} ;

```