```
   Help
#include "hullwhite2d_stdi.h"
#include "pnl/pnl_vector.h"
#include "pnl/pnl_matrix.h"
#include "math/InterestRateModelTree/TreeHW2D/TreeHW2D.h"
#include "hullwhite2d_includes.h"

//The "#else" part of the code will be freely available aft
    er the (year of creation of this file + 2)
#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 th
    e tree (see TreeHW2D.h)
/// ModelHW2D    : structure that contains  the parameters
    of the Hull&White one factor model (see TreeHW2D.h)
/// ZCMarketData : structure that contains the Zero Coupon
    Bond prices of the market, or given by a constant yield-to-
    maturity (see InitialYieldCurve.h)

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

    jminprev = pnl_vect_int_get(Meth->yIndexMin, Meth->Ng
    rid);  // jmin(Ngrid)
    jmaxprev = pnl_vect_int_get(Meth->yIndexMax, Meth->Ng
```

```
    rid);  // jmax(Ngrid)
    kminprev = pnl_vect_int_get(Meth->uIndexMin, Meth->Ng
    rid);  // kmin(Ngrid)
    kmaxprev = pnl_vect_int_get(Meth->uIndexMax, Meth->Ng
    rid);  // kmax(Ngrid)

    pnl_mat_resize(OptionPriceMat2, jmaxprev-jminprev+1, km
    axprev-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* ZCMarket, double T, int NumberOfTimeS
    tep,double r, double u)
{
    double a ,sigma1, b, sigma2, rho;
    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*sigma2 / (b-a) );
```

```
    //rho_y_u = (rho * sigma1 + sigma2/(b-a)) / sigma3;

    ///****************** Computation of the vector of payo
    ff at the maturity of the option ******************///
    ZCBond_InitialPayoffHW2D(Meth, OptionPriceMat2);

    ///****************** Computation of the vector of payo
    ff at the maturity of the option ******************///
    BackwardIterationHW2D(Meth, ModelParam, ZCMarket,
    OptionPriceMat1, OptionPriceMat2, Meth->Ngrid, 0);

    ///****************** 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,double u0,
    double a,double sigma1,double b,double sigma2,double rho,
    double T,int N_steps,double *price)
{
  TreeHW2D Tr;
  ModelHW2D ModelParams;
  ZCMarketData ZCMarket;

  /* Flag to decide to read or not ZC bond datas in "initia
    lyields.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;
    ReadMarketData(&ZCMarket);

    if(T > GET(ZCMarket.tm,ZCMarket.Nvalue-1))
    {
        printf("{nError : time bigger than the last time
  value entered in initialyield.dat{n");
        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 {"Speed of Mean Reversion of u{" must be diffe
  rents! {n");
    exit(EXIT_FAILURE);
}

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

// Construction of the tree, calibrated to the initial yi
  eld 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),
                      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},FO
    RBID} */,{" ",PREMIA_NULLTYPE,{0},FORBID}},
  CHK_OPT(TR_ZCBONDHW2D),
  CHK_ok,
  MET(Init)
} ;
```

References