```
   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_CAPFLOORHW2D)(void *Opt,void *Mod,Pricing
    Method *Met)
{
return AVAILABLE_IN_FULL_PREMIA;
}
static int CHK_OPT(TR_CAPFLOORHW2D)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
#else

/// Computation of the payoff of a caplet/floorlet paying
    at T1 max[0, 1-P(T1,T2)*(1+CapFloorFixedRate*(T2-T1))] (for
    a caplet)
static void CapFloor_InitialPayoff(TreeHW2D* Meth, ModelHW2
    D* ModelParam, ZCMarketData* ZCMarket, PnlMat* OptionPriceM
    at2, NumFunc_1 *p, double T1, double T2, double CapFloorFix
    edRate)
{
    double a ,sigma1, b, sigma2, rho, sigma3;

    int jminprev, jmaxprev, kminprev, kmaxprev; // jmin[i],
     jmax [i]
    int j, k; // i = represents the time index. h, l, j, k
    represents the nodes index

    double delta_y2; // delta_y1 = space step of the proces
    s y at time i ; delta_y2 same at time i+1.
    double delta_u2; // delta_u1 = space step of the proces
    s u at time i ; delta_u2 same at time i+1.
```

```
double delta_t1; // time step

double ZCPrice;
double current_rate, current_u;
int  i_T1;
double periodicity;


/// 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 payoff at the maturit
y of the option
periodicity = T2 - T1;
i_T1 = indiceTimeHW2D(Meth, T1);

jminprev = pnl_vect_int_get(Meth->yIndexMin, i_T1);  //
 jmin(i_T1)
jmaxprev = pnl_vect_int_get(Meth->yIndexMax, i_T1);  //
 jmax(i_T1)
kminprev = pnl_vect_int_get(Meth->uIndexMin, i_T1);  //
 kmin(i_T1)
kmaxprev = pnl_vect_int_get(Meth->uIndexMax, i_T1);  //
 kmax(i_T1)

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

delta_t1 = GET(Meth->t, i_T1) - GET(Meth->t, i_T1-1); /
/ Pas de temps entre t[Ngrid-1] et t[Ngrid]
delta_y2 = delta_xHW2D(delta_t1, a, sigma3); // delta_
```

```
    y (Ngrid)
    delta_u2 = delta_xHW2D(delta_t1, b, sigma2); // delta_
    u (Ngrid)

    p->Par[0].Val.V_DOUBLE = 1.0 ;

    for( j = jminprev ; j<=jmaxprev ; j++)
    {
        for( k = kminprev ; k<=kmaxprev ; k++)
        {
            current_u = k * delta_u2;
            current_rate = j * delta_y2 - current_u/(b-a) +
     GET(Meth->alpha, i_T1); // rate(Ngrid,j, k)

            ZCPrice = cf_hw2d_zcb(ZCMarket, a, sigma1, b,
    sigma2, rho, T1, current_rate, current_u, T2);

            MLET(OptionPriceMat2, j-jminprev, k-kminprev) =
     (p->Compute)(p->Par, (1+periodicity*CapFloorFixedRate)*
    ZCPrice);
        }
    }

}

/// Prce of a Cap/Floor using a trinomial tree
static double tr_hw2d_capfloor(TreeHW2D* Meth, ModelHW2D*
    ModelParam, ZCMarketData* ZCMarket, int NumberOfTimeStep,
    NumFunc_1 *p, double r, double u, double periodicity,double fir
    st_reset_date,double contract_maturity, double CapFloorFixed
    Rate)
{
    double a ,sigma1, b, sigma2, rho;
    double OptionPrice, Ti2, Ti1;
    int i, i_Ti2, i_Ti1, Number_of_Payements;

    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;


///**************** PAYOFF at the MATURITY of the
OPTION : T(n-1)****************///
Ti2 = contract_maturity;
Ti1 = Ti2 - periodicity;

// Computation in the Matrix OptionPriceMat2 of the
price at Ti1 of the last caplet/floorlet
CapFloor_InitialPayoff(Meth, ModelParam, ZCMarket,
OptionPriceMat2, p, Ti1, Ti2, CapFloorFixedRate);


///**************** Backward computation of the option
price ****************///
Number_of_Payements = (int) ((contract_maturity-first_
reset_date)/periodicity);

for(i = Number_of_Payements-2; i>=0; i--)
{
    Ti1 = first_reset_date + i * periodicity; // T(i)
    Ti2 = Ti1 + periodicity;                  // T(i+1)
    i_Ti2 = indiceTimeHW2D(Meth, Ti2);
    i_Ti1 = indiceTimeHW2D(Meth, Ti1);
```

```
        // Backward computation from T(i+1) to T(i)
        BackwardIterationHW2D(Meth, ModelParam, ZCMarket,
    OptionPriceMat1, OptionPriceMat2, i_Ti2, i_Ti1);

        // Computation in the Matrix OptionPriceMat1 of th
    e price at T(i) of the last caplet/floorlet
        CapFloor_InitialPayoff(Meth, ModelParam, ZCMarket,
    OptionPriceMat1, p, Ti1, Ti2, CapFloorFixedRate);

        // Add the price of the caplet/floorlet to the
    price of the Cap/Floor
        pnl_mat_plus_mat(OptionPriceMat2, OptionPriceMat1);

    }
    ///****************** Backward computation of the
    option price from  first_reset_date to 0 ********************///
    i_Ti2 = indiceTimeHW2D(Meth, first_reset_date);
    i_Ti1 = 0;
    BackwardIterationHW2D(Meth, ModelParam, ZCMarket,
    OptionPriceMat1, OptionPriceMat2, i_Ti2, i_Ti1);

    OptionPrice = MGET(OptionPriceMat1, 0, 0);

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

    return OptionPrice;

}// FIN de la fonction ZCOption


static int tr_capfloor2d(int flat_flag, double r0,double u0
    ,double a,double sigma1,double b,double sigma2,double rho,
     double contract_maturity, double first_reset_date,
    double periodicity,double Nominal, double CapFloorFixedRate,
    NumFunc_1 *p, long NtY, 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(contract_maturity > GET(ZCMarket.tm,ZCMarket.Nvalu
  e-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 TimeGride from 0 to T(n-1)=contrac
  t_maturity-periodicity
SetTimegridHW2D_Cap(&Tr , NtY, first_reset_date, contrac
  t_maturity-periodicity, periodicity);
```

```
  // Construction of the tree, calibrated to the initial yi
    eld curve
  SetTreeHW2D(&Tr, &ModelParams, &ZCMarket);

  *price =  Nominal * tr_hw2d_capfloor(&Tr, &ModelParams, &
    ZCMarket, NtY, p, r0, u0, periodicity, first_reset_date,
    contract_maturity, CapFloorFixedRate);

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

  return OK;
}



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

int CALC(TR_CAPFLOORHW2D)(void *Opt,void *Mod,Pricing
    Method *Met)
{
  TYPEOPT* ptOpt=(TYPEOPT*)Opt;
  TYPEMOD* ptMod=(TYPEMOD*)Mod;

  return  tr_capfloor2d(  ptMod->flat_flag.Val.V_INT,
                          MOD(GetYield)(ptMod),
                          ptMod->InitialYieldsu.Val.V_PDOUB
    LE,
                          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-pt
    Mod->T.Val.V_DATE,
                          ptOpt->FirstResetDate.Val.V_DATE-
    ptMod->T.Val.V_DATE,
                          ptOpt->ResetPeriod.Val.V_DATE,
                          ptOpt->Nominal.Val.V_PDOUBLE,
```

```
                          ptOpt->FixedRate.Val.V_PDOUBLE,
                          ptOpt->PayOff.Val.V_NUMFUNC_1,
                          Met->Par[0].Val.V_LONG,
                          &(Met->Res[0].Val.V_DOUBLE));
}
static int CHK_OPT(TR_CAPFLOORHW2D)(void *Opt, void *Mod)
{
  if ((strcmp(((Option*)Opt)->Name,"Cap")==0) || (strcmp(((
    Option*)Opt)->Name,"Floor")==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=10;


    }
  return OK;
}


PricingMethod MET(TR_CAPFLOORHW2D)=
{
  "TR_CAPFLOORHW2D",
  {{"TimeStepNumber for Period",LONG,{100},ALLOW},
   {" ",PREMIA_NULLTYPE,{0},FORBID}},
  CALC(TR_CAPFLOORHW2D),
  {{"Price",DOUBLE,{100},FORBID}/*,{"Delta",DOUBLE,{100},FO
    RBID} */,{" ",PREMIA_NULLTYPE,{0},FORBID}},
  CHK_OPT(TR_CAPFLOORHW2D),
  CHK_ok,
  MET(Init)
```

```
} ;
```

References