```
  Help
#include <stdlib.h>
#include "hullwhite1d_stdi.h"
#include "math/InterestRateModelTree/TreeShortRate/TreeShortRate.h"
#include "pnl/pnl_vector.h"
#include "hullwhite1d_includes.h"



#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2007+2) //The "#else" part of the code will be freely available after the (year of creation of this file + 2)
static int CHK_OPT(TR_CapFloorHW1D)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(TR_CapFloorHW1D)(void *Opt,void *Mod,PricingMethod *Met)
{
return AVAILABLE_IN_FULL_PREMIA;
}
#else

/// Computation of the payoff at the final time of the tree (ie the option maturity)
static void CapFloor_InitialPayoffHW1D(TreeShortRate* Meth, ModelParameters* ModelParam, ZCMarketData* ZCMarket, PnlVect* OptionPriceVect2, NumFunc_1 *p, double T1, double T2, double CapFloorFixedRate)
{
  double a,sigma;
    int jminprev, jmaxprev; // jmin[i], jmax [i]
    int j; //  j represents the nodes index

    double delta_x2; // delta_x1 = space step of the process x at time i ; delta_x2 same at time i+1.
    double delta_t1; // time step

    double ZCPrice;
    double current_rate;
    int  i_T1;
    double periodicity;


    /// Parameters of the process r
    a = (ModelParam->MeanReversion);
    sigma = (ModelParam->RateVolatility);

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

    jminprev = pnl_vect_int_get(Meth->Jminimum, i_T1);  //
    jmin(i_T1)
    jmaxprev = pnl_vect_int_get(Meth->Jmaximum, i_T1);  //
    jmax(i_T1)

    pnl_vect_resize(OptionPriceVect2, jmaxprev-jminprev+1);

    delta_t1 = GET(Meth->t, i_T1) - GET(Meth->t, i_T1-1);
    delta_x2 = SpaceStep(delta_t1, a, sigma); // delta_x (
    i_T1)

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

    for( j = jminprev ; j<=jmaxprev ; j++)
    {
        current_rate = func_model_hw1d(j * delta_x2  + GET(
    Meth->alpha, i_T1)); // rate(Ngrid,j, k)

        ZCPrice = cf_hw1d_zcb(ZCMarket, a, sigma, T1,
    current_rate, T2);

        LET(OptionPriceVect2, j-jminprev) = (p->Compute)(p-
    >Par, (1+periodicity*CapFloorFixedRate)*ZCPrice);
    }

}

/// Price of a Cap/Floor using a trinomial tree
```

```
static double tr_hw1d_capfloor(TreeShortRate* Meth, ModelP
    arameters* ModelParam, ZCMarketData* ZCMarket, int NumberO
    fTimeStep, NumFunc_1 *p, double r, double periodicity,
    double first_reset_date,double contract_maturity, double CapF
    loorFixedRate)
{
    double OptionPrice, Ti2, Ti1;
    int i, i_Ti2, i_Ti1, n;

    PnlVect* OptionPriceVect1; // Vector of prices of the
    option at i
    PnlVect* OptionPriceVect2; // Vector of prices of the
    option at i+1
    OptionPriceVect1 = pnl_vect_create(1);
    OptionPriceVect2 = pnl_vect_create(1);


    ///*******************Parameters of the process r *****
    ****************////
    //a = ModelParam->MeanReversion;
    //sigma = ModelParam->RateVolatility;

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

    CapFloor_InitialPayoffHW1D(Meth, ModelParam, ZCMarket,
    OptionPriceVect2, p, Ti1, Ti2, CapFloorFixedRate);

    ///**************** Backward computation of the option
    price ***************///
    n = (int) ((contract_maturity-first_reset_date)/perio
    dicity + 0.1);

    for(i = n-2; i>=0; i--)
    {
        Ti1 = first_reset_date + i * periodicity;
        Ti2 = Ti1 + periodicity;
        i_Ti2 = IndexTime(Meth, Ti2);
        i_Ti1 = IndexTime(Meth, Ti1);
        BackwardIteration(Meth, ModelParam, OptionPriceVec
    t1, OptionPriceVect2, i_Ti2, i_Ti1,&func_model_hw1d);

        CapFloor_InitialPayoffHW1D(Meth, ModelParam, ZCMar
    ket, OptionPriceVect1, p, Ti1, Ti2, CapFloorFixedRate);

        pnl_vect_plus_vect(OptionPriceVect2, OptionPriceVec
    t1);
    }

    ///****************** Backward computation of the
    option price from  first_reset_date to 0 *******************///
    i_Ti2 = IndexTime(Meth, first_reset_date);
    i_Ti1 = 0;
    BackwardIteration(Meth, ModelParam, OptionPriceVect1,
    OptionPriceVect2, i_Ti2, i_Ti1, &func_model_hw1d);

    OptionPrice = GET(OptionPriceVect1, 0);

    pnl_vect_free(& OptionPriceVect1);
    pnl_vect_free(& OptionPriceVect2);

    return OptionPrice;

}



static int tr_capfloor1d(int flat_flag,double r0,double a,
    double sigma, double contract_maturity, double first_reset_date,
     double periodicity,double Nominal, double CapFloorFixedRa
    te, NumFunc_1 *p, long N_steps, double *price)
{
    TreeShortRate Tr;
    ModelParameters ModelParams;
    ZCMarketData ZCMarket;

    /* Flag to decide to read or not ZC bond datas in "ini
    tialyields.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.Nv
    alue-1))
        {
            printf("{nError : time bigger than the last
    time value entered in initialyield.dat{n");
            exit(EXIT_FAILURE);
        }
    }

    ModelParams.MeanReversion  = a;
    ModelParams.RateVolatility = sigma;

    // Construction of the Time Grid
    SetTimeGrid_Tenor(&Tr, N_steps, first_reset_date, contr
    act_maturity-periodicity, periodicity);

    // Construction of the tree, calibrated to the initial
    yield curve
    SetTreeShortRate(&Tr, &ModelParams, &ZCMarket, &func_
    model_hw1d, &func_model_der_hw1d, &func_model_inv_hw1d);

    *price =  Nominal * tr_hw1d_capfloor(&Tr, &ModelParams,
     &ZCMarket, N_steps, p, r0, periodicity, first_reset_date,
     contract_maturity, CapFloorFixedRate);

    DeleteTreeShortRate(&Tr);
    DeleteZCMarketData(&ZCMarket);

  return OK;
}
```

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

int CALC(TR_CapFloorHW1D)(void *Opt,void *Mod,Pricing
    Method *Met)
{
  TYPEOPT* ptOpt=(TYPEOPT*)Opt;
  TYPEMOD* ptMod=(TYPEMOD*)Mod;

  return  tr_capfloor1d(    ptMod->flat_flag.Val.V_INT,
                            MOD(GetYield)(ptMod),
                            ptMod->a.Val.V_DOUBLE,
                            ptMod->Sigma.Val.V_PDOUBLE,
                            ptOpt->BMaturity.Val.V_DATE-pt
    Mod->T.Val.V_DATE,
                            ptOpt->FirstResetDate.Val.V_DA
    TE-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_CapFloorHW1D)(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_INT=10;
    }
  return OK;
}


PricingMethod MET(TR_CapFloorHW1D)=
{
  "TR_HullWhite1d_CapFloor",
  { {"TimeStepNumber per Period",INT,{100},ALLOW},
    {" ",PREMIA_NULLTYPE,{0},FORBID}},
  CALC(TR_CapFloorHW1D),
  { {"Price",DOUBLE,{100},FORBID},
    {" ",PREMIA_NULLTYPE,{0},FORBID}},
  CHK_OPT(TR_CapFloorHW1D),
  CHK_ok,
  MET(Init)
} ;
```

References