Help

```
#include "hullwhite1dgeneralized_stdi.h"

#include "pnl/pnl_mathtools.h"
#include "pnl/pnl_vector.h"

#include "math/InterestRateModelTree/TreeHW1dGeneralized/TreeHW1dGeneralized.h"
#include "math/read_market_zc/InitialYieldCurve.h"

//The "#else" part of the code will be freely available after the (year of creation of this file + 2)
#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2010+2)
static int CHK_OPT(TR_BermudianSwaptionHW1DG)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(TR_BermudianSwaptionHW1DG)(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 BermudianSwaption_InitialPayoffHW1D(int swaption_start, TreeHW1dG* Meth, ModelHW1dG* HW1dG_Parameters, ZCMarketData* ZCMarket, PnlVect* OptionPriceVect2, NumFunc_1 *p, double periodicity,double contract_maturity, double SwaptionFixedRate)
{
    double sigma;

    int jminprev, jmaxprev; // jmin[i], jmax [i]
    int i,j;
```

```
double delta_x1; // delta_x1 = space step of the proces
s x at time i
double delta_t1; // time step

double ZCPrice, SumZC;
double current_rate;

int NumberOfPayments;
double Ti;

ZCPrice = 0.0;

///** Calcul du vecteur des payoffs a l'instant de matu
rite de l'option
jminprev = pnl_vect_int_get(Meth->Jminimum, swaption_
start);   // jmin(swaption_start)
jmaxprev = pnl_vect_int_get(Meth->Jmaximum, swaption_
start);  // jmax(swaption_start)

pnl_vect_resize(OptionPriceVect2, jmaxprev-jminprev+1);

delta_t1 = GET(Meth->t, swaption_start) - GET(Meth->t,
swaption_start-1);
sigma = Current_VolatilityHW1dG(HW1dG_Parameters, GET(
Meth->t, swaption_start));
delta_x1 = SpaceStepHW1dG(delta_t1, sigma);//SpaceStep    HW1dG(delta_t1, a,

NumberOfPayments = (int) floor((contract_maturity-GET(
Meth->t, swaption_start) )/periodicity + 0.2);


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

for( j = jminprev ; j<=jmaxprev ; j++)
{
    current_rate = j * delta_x1 + GET(Meth->alpha, swa
ption_start); // rate(Ngrid, j )

    SumZC = 0;
    for(i=1; i<=NumberOfPayments; i++)
    {
            Ti = GET(Meth->t, swaption_start) + i*periodic
    ity;
            ZCPrice = DiscountFactor(ZCMarket, HW1dG_Para
    meters, GET(Meth->t, swaption_start), Ti, current_rate);

            SumZC += ZCPrice;
        }


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

}

/// Price of a bermudianswaption using a trinomial tree
static double tr_hw1dg_bermudianswaption(TreeHW1dG* Meth,
    ModelHW1dG* HW1dG_Parameters, ZCMarketData* ZCMarket,int Numb
    erOfTimeStep, NumFunc_1 *p, double r, double periodicity,
    double option_maturity,double contract_maturity, double Swaptio
    nFixedRate)
{
    double delta_t1; // time step
    double Pup, Pmiddle, Pdown;
    int i,j;
    double Ti2, Ti1;
    int i_Ti2, i_Ti1;
    double current_rate, NumberOfPayments;
    double OptionPrice;

    PnlVect* PayoffVect;
    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);
    PayoffVect = pnl_vect_create(1);

    //mean_reversion = (HW1dG_Parameters->MeanReversion);
```

```
///****************** Computation of the vector of payo
ff at the maturity of the option ******************///
Ti1 = contract_maturity-periodicity;
i_Ti1 = IndexTimeHW1dG(Meth, Ti1);
BermudianSwaption_InitialPayoffHW1D(i_Ti1, Meth, HW1dG_
Parameters, ZCMarket, OptionPriceVect2, p, periodicity, contr
act_maturity, SwaptionFixedRate);

///****************** Backward computation of the
option price until initial time s ******************///

NumberOfPayments = (int) floor((contract_maturity-
option_maturity )/periodicity + 0.2);

for(i=NumberOfPayments-2 ; i>=0 ; i--)
{
    Ti1 = option_maturity + i * periodicity;
    Ti2 = Ti1 + periodicity;
    i_Ti2 = IndexTimeHW1dG(Meth, Ti2);
    i_Ti1 = IndexTimeHW1dG(Meth, Ti1);

    BackwardIterationHW1dG(Meth, HW1dG_Parameters,
OptionPriceVect1, OptionPriceVect2, i_Ti2, i_Ti1);

    BermudianSwaption_InitialPayoffHW1D(i_Ti1, Meth,     HW1dG_Parameters, Z
act_maturity, SwaptionFixedRate);

    for(j=0;j<PayoffVect->size;j++)
    {
        if(GET(PayoffVect,j)>GET(OptionPriceVect2,j))
        {
            LET(OptionPriceVect2,j)=GET(PayoffVect,j);
        }
    }

}

BackwardIterationHW1dG(Meth, HW1dG_Parameters, OptionP
riceVect1, OptionPriceVect2, i_Ti1, 1);
```

```
    Pup = 1.0 / 6.0;
    Pmiddle = 2.0 /3.0 ;
    Pdown = 1.0 / 6.0;

    delta_t1 = GET(Meth->t, 1) - GET(Meth->t,0);
    current_rate = GET(Meth->alpha, 0); // r(0,j)
    OptionPrice = exp(-current_rate*delta_t1) * ( Pup * GET
    (OptionPriceVect2, 2) + Pmiddle * GET(OptionPriceVect2,1)
    + Pdown * GET(OptionPriceVect2, 0));

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

    return OptionPrice;

}


static int tr_bermudianswaption1d(int flat_flag, double r0,
     int CapletCurve, double a, double contract_maturity,
    double option_maturity, double periodicity,double Nominal,
    double SwaptionFixedRate, NumFunc_1 *p, int N_steps, double *
    price)
{
    TreeHW1dG Tr;
    ModelHW1dG HW1dG_Parameters;
    ZCMarketData ZCMarket;
    MktATMCapletVolData MktATMCapletVol;

    // Read the interest rate term structure from file, or
    set it flat
    if(flat_flag==0)
    {
        ZCMarket.FlatOrMarket = 0;
        ZCMarket.Rate = r0;
    }

    else
    {
        ZCMarket.FlatOrMarket = 1;
        ReadMarketData(&ZCMarket);
    }

    // Read the caplet volatilities from file "impliedcapl
    etvol.dat".
    ReadCapletMarketData(&MktATMCapletVol, CapletCurve);

    hw1dg_calibrate_volatility(&HW1dG_Parameters, &ZCMarke
    t, &MktATMCapletVol, a);

    // Construction of the Time Grid
    SetTimeGrid_TenorHW1dG(&Tr, N_steps, option_maturity,
    contract_maturity, periodicity);

    // Construction of the tree, calibrated to the initial
    yield curve
    SetTreeHW1dG(&Tr, &HW1dG_Parameters, &ZCMarket);

    *price = Nominal * tr_hw1dg_bermudianswaption(&Tr, &    HW1dG_Parameters, &Z
    option_maturity, contract_maturity, SwaptionFixedRate);

    DeleteTreeHW1dG(&Tr);
    DeleteZCMarketData(&ZCMarket);
    DeleteMktATMCapletVolData(&MktATMCapletVol);
    DeletModelHW1dG(&HW1dG_Parameters);

    return OK;
}


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

int CALC(TR_BermudianSwaptionHW1DG)(void *Opt,void *Mod,
    PricingMethod *Met)
{
  TYPEOPT* ptOpt=(TYPEOPT*)Opt;
  TYPEMOD* ptMod=(TYPEMOD*)Mod;

  return  tr_bermudianswaption1d(   ptMod->flat_flag.Val.V_
    INT,
                                    MOD(GetYield)(ptMod),
                                    ptMod->CapletCurve.Val.
    V_ENUM.value,
                                    ptMod->a.Val.V_DOUBLE,
                                    ptOpt->BMaturity.Val.V_
    DATE-ptMod->T.Val.V_DATE,
                                    ptOpt->OMaturity.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_BermudianSwaptionHW1DG)(void *Opt,
    void *Mod)
{
    if ((strcmp(((Option*)Opt)->Name,"PayerBermudanSwaptio
    n")==0) || (strcmp(((Option*)Opt)->Name,"
    ReceiverBermudanSwaption")==0))
        return OK;
    else
        return WRONG;
}
#endif //PremiaCurrentVersion


static int MET(Init)(PricingMethod *Met,Option *Opt)
{
  if ( Met->init == 0)
    {
      Met->init=1;
       Met->HelpFilenameHint = "    tr_hullwhite1dgeneralized_bermudianswaption"
      Met->Par[0].Val.V_INT=50;
    }
  return OK;
}

PricingMethod MET(TR_BermudianSwaptionHW1DG)=
{
  "TR_HullWhite1dG_BermudianSwaption",
  {{"TimeStepNumber per Period",INT,{100},ALLOW},
   {" ",PREMIA_NULLTYPE,{0},FORBID}},
  CALC(TR_BermudianSwaptionHW1DG),
  {{"Price",DOUBLE,{100},FORBID}/*,{"Delta",DOUBLE,{100},FO
    RBID}*/ ,{" ",PREMIA_NULLTYPE,{0},FORBID}},
  CHK_OPT(TR_BermudianSwaptionHW1DG),
  CHK_ok,
  MET(Init)
} ;
```

References