

[Help](#)

```

#include <stdlib.h>
#include "hullwhite1d_std.h"
#include "math/InterestRateModelTree/TreeShortRate/TreeShortRate.h"
#include "pnl/pnl_vector.h"

//The "#else" part of the code will be freely available after the (year of creat
#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2007+2)
static int CHK_OPT(TR_SwaptionHW1D)(void *Opt, void *Mod)
{
    return NONACTIVE;
}
int CALC(TR_SwaptionHW1D)(void *Opt, void *Mod, PricingMethod *Met)
{
    return AVAILABLE_IN_FULL_PREMIA;
}
#else

/// TreeShortRate      : structure that contains components of the tree (see Tree
/// ModelParameters    : structure that contains the parameters of the Hull&Whi
/// 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 option matur
void Swaption_InitialPayoffHW1D(TreeShortRate *Meth, ModelParameters *ModelParam
{
    double a , sigma;

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

    double delta_x1; // delta_x1 = space step of the process x at time i
    double delta_t1; // time step

    double ZCPrice, SumZC;
    double current_rate;

    int NumberOfPayments;
    double Ti;

```

```

ZCPrice = 0.; /* to avoid warning */
///<*****Parameters of the process r *****/
a = ModelParam->MeanReversion;
sigma = ModelParam->RateVolatility;

///<** Calcul du vecteur des payoffs a l'instant de maturite de l'option
jminprev = pnl_vect_int_get(Meth->Jminimum, Meth->Ngrid); // jmin(Ngrid)
jmaxprev = pnl_vect_int_get(Meth->Jmaximum, Meth->Ngrid); // jmax(Ngrid)

pnl_vect_resize(OptionPriceVect2, jmaxprev - jminprev + 1);

delta_t1 = GET(Meth->t, Meth->Ngrid) - GET(Meth->t, Meth->Ngrid - 1); // Pas d
delta_x1 = SpaceStep(delta_t1, a, sigma); // SpaceStep(Ngrid)

NumberOfPayments = (int) floor((contract_maturity - option_maturity) / periodi

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

for (j = jminprev ; j <= jmaxprev ; j++)
{
    current_rate = func_model_hw1d(j * delta_x1 + GET(Meth->alpha, Meth->Ngrid

    SumZC = 0;
    for (i = 1; i <= NumberOfPayments; i++)
    {
        Ti = option_maturity + i * periodicity;
        ZCPrice = cf_hw1d_zcb(ZCMarket, a, sigma, option_maturity, current_rat

        SumZC += ZCPrice;
    }

    //SwapRate = (1-ZCPrice) / (periodicity*SumZC);

    LET(OptionPriceVect2, j - jminprev) = ((p->Compute)(p->Par, periodicity *

    //LET(OptionPriceVect2, j-jminprev) = SumZC* periodicity*(p->Compute)(p->P
}
}

```

```

/// Price of a swaption using a trinomial tree
double tr_hw1d_swaption(TreeShortRate *Meth, ModelParameters *ModelParam, ZCMarketData *ZCMarket)
{
    int index_last, index_first;
    double OptionPrice;

    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 processes r, u and y *****
    //a = ModelParam->MeanReversion;
    //sigma = ModelParam->RateVolatility;

    ///***** Computation of the vector of payoff at the maturity of the swaption *****
    Swaption_InitialPayoffHW1D(Meth, ModelParam, ZCMarket, OptionPriceVect2, p, pe);

    ///***** Backward computation of the option price until initial time *****
    index_last = Meth->Ngrid;
    index_first = 0;

    BackwardIteration(Meth, ModelParam, OptionPriceVect1, OptionPriceVect2, index_last, index_first);

    OptionPrice = GET(OptionPriceVect1, 0);

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

    return OptionPrice;
}

static int tr_swaption1d(int flat_flag, double r0, char *curve, double a, double sigma)
{
    TreeShortRate Tr;
    ModelParameters 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 (option_maturity > GET(ZCMarket.tm, ZCMarket.Nvalue - 1))
    {
        printf("\ nError : time bigger than the last time value entered in ini
        exit(EXIT_FAILURE);
    }
}

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

// Construction of the Time Grid
SetTimeGrid(&Tr, N_steps, option_maturity);

// Construction of the tree, calibrated to the initial yield curve
SetTreeShortRate(&Tr, &ModelParams, &ZCMarket, &func_model_hw1d, &func_model_d

*price = Nominal * tr_hw1d_swaption(&Tr, &ModelParams, &ZCMarket, N_steps, p,

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

return OK;
}

///  

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

int CALC(TR_SwaptionHW1D)(void *Opt, void *Mod, PricingMethod *Met)

```

```

{
    TYPEOPT *ptOpt = (TYPEOPT *)Opt;
    TYPEMOD *ptMod = (TYPEMOD *)Mod;

    return tr_swaption1d(ptMod->flat_flag.Val.V_INT,
                        MOD(GetYield)(ptMod),
                        MOD(GetCurve)(ptMod),
                        ptMod->a.Val.V_DOUBLE,
                        ptMod->Sigma.Val.V_PDOUBLE,
                        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_SwaptionHW1D)(void *Opt, void *Mod)
{
    if ((strcmp(((Option *)Opt)->Name, "PayerSwaption") == 0) || (strcmp(((Option
        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 = 200;
    }

    return OK;
}

PricingMethod MET(TR_SwaptionHW1D) =
{

```

```
"TR_HullWhite1d_Swaption",
{ {"TimeStepNumber", LONG, {100}, ALLOW},
  {" ", PREMIA_NULLTYPE, {0}, FORBID}
},
CALC(TR_SwaptionHW1D),
{{"Price", DOUBLE, {100}, FORBID}/*,{"Delta",DOUBLE,{100},FORBID}*/ , {" ", PR
CHK_OPT(TR_SwaptionHW1D),
CHK_ok,
MET(Init)
} ;
```