

Help

```
#include "hullwhite1dgeneralized_std.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 creat
#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2010+2)
static int CHK_OPT(TR_ZCBondHW1dG)(void *Opt, void *Mod)
{
    return NONACTIVE;
}
int CALC(TR_ZCBondHW1dG)(void *Opt, void *Mod, PricingMethod *Met)
{
    return AVAILABLE_IN_FULL_PREMIA;
}
#else

/// TreeHW1dG      : structure that contains components of the tree (see TreeHW1d
/// ModelHW1dG     : structure that contains the parameters of the Hull&White on
/// 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 ZCBond matur
void ZCBond_InitialPayoffHW1dG(TreeHW1dG *Meth, PnlVect *OptionPriceVect2)
{
    int jminprev, jmaxprev;

    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);

    pnl_vect_set_double(OptionPriceVect2, 1.0); // Payoff = 1 for a ZC bond
}

// Price at time "s" of a ZC bond maturing at "T" using a trinomial tree.
double tr_hw1dg_zcbond(TreeHW1dG *Meth, ModelHW1dG *HW1dG_Parameters, ZCMarketDa
```

```

{

    double delta_t1; // time step

    double current_rate;
    double Pup, Pdown, Pmiddle;
    double OptionPrice;

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

    ///***** Computation of the vector of payoff at the maturity of t
    ZCBond_InitialPayoffHW1dG(Meth, OptionPriceVect2);

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

    BackwardIterationHW1dG(Meth, HW1dG_Parameters, OptionPriceVect1, OptionPriceVect2);

    // First node of the tree
    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); // Pas de temps entre t[1] et t[0]
    current_rate = GET(Meth->alpha, 0); // r(i,j)
    OptionPrice = exp(-current_rate * delta_t1) * (Pup * GET(OptionPriceVect1, 2) +
    Pmiddle * GET(OptionPriceVect1, 1) + Pdown * GET(OptionPriceVect1, 0));

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

    return OptionPrice;

}

// Price
int tr_zcbond1d(int flat_flag, double r0, char *curve, int CapletCurve, double a)
{
    TreeHW1dG Tr;

```

```
ModelHW1dG ModelParams;
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;
    ZCMarket.filename = curve;
    ReadMarketData(&ZCMarket);
}

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

hw1dg_calibrate_volatility(&ModelParams, &ZCMarket, &MktATMCapletVol, a);

// Construction of the Time Grid
SetTimeGridHW1dG(&Tr, N_steps, 0, T);

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

*price = tr_hw1dg_zcbond(&Tr, &ModelParams, &ZCMarket, T);

DeleteTreeHW1dG(&Tr);
DeleteZCMarketData(&ZCMarket);
DeleteMktATMCapletVolData(&MktATMCapletVol);
DeleteModelHW1dG(&ModelParams);

return 0;
}
```

```

//***** PREMIA FUNCTIONS *****
int CALC(TR_ZCBondHW1dG)(void *Opt, void *Mod, PricingMethod *Met)
{
    TYPEOPT *ptOpt = (TYPEOPT *)Opt;
    TYPEMOD *ptMod = (TYPEMOD *)Mod;

    return tr_zcbond1d(ptMod->flat_flag.Val.V_INT,
                      MOD(GetYield)(ptMod),
                      MOD(GetCurve)(ptMod),
                      ptMod->CapletCurve.Val.V_ENUM.value,
                      ptMod->a.Val.V_DOUBLE,
                      ptOpt->BMaturity.Val.V_DATE - ptMod->T.Val.V_DATE,
                      Met->Par[0].Val.V_INT,
                      &(Met->Res[0].Val.V_DOUBLE));
}

static int CHK_OPT(TR_ZCBondHW1dG)(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->HelpFilenameHint = "tr_hullwhite1dgeneralized_zcbond";
        Met->Par[0].Val.V_LONG = 100;
    }
    return OK;
}

PricingMethod MET(TR_ZCBondHW1dG) =
{
    "TR_HullWhite1dG_ZCBond",

```

```
{ {"TimeStepNumber", LONG, {100}, ALLOW},
  {" ", PREMIA_NULLTYPE, {0}, FORBID}
},
CALC(TR_ZCBondHW1dG),
{{"Price", DOUBLE, {100}, FORBID}, {" ", PREMIA_NULLTYPE, {0}, FORBID}},
CHK_OPT(TR_ZCBondHW1dG),
CHK_ok,
MET(Init)
} ;
```