```
   Help
#include "cirpp1d_stdi.h"
#include "pnl/pnl_vector.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 <
     (2007+2)
static int CHK_OPT(TR_ZCBondCIRpp1D)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(TR_ZCBondCIRpp1D)(void *Opt,void *Mod,Pricing
    Method *Met)
{
return AVAILABLE_IN_FULL_PREMIA;
}
#else

/// TreeCIRpp1D     : structure that contains components of
     the tree (see TreeCIRpp1D.h)
/// ModelCIRpp1D    : structure that contains  the paramete
    rs of the Hull&White one factor model (see TreeCIRpp1D.h)
/// ZCMarketData : structure that contains the Zero Coupon
    Bond prices of the market, or given by a constant yield-to-
    maturity (see InitialYieldCurve.h)

/// Computation of the payoff at the final time of the tre
    e (ie the ZCBond maturity)
static void ZCBond_InitialPayoffCIRpp1D(TreeCIRpp1D* Meth,
    PnlVect* OptionPriceVect2)
{
    int NumberNode;

    NumberNode = (int) ((GET(Meth->Xmax, Meth->Ngrid) - GET
    (Meth->Xmin, Meth->Ngrid)) / (Meth->delta_x) + 0.1);

    pnl_vect_resize(OptionPriceVect2, NumberNode+1);

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

```
}

/// Backward computation of the price of a Zero Coupon Bond
static void ZCBond_BackwardIterationCIRpp1D(TreeCIRpp1D*
    Meth, ModelCIRpp1D* ModelParam, ZCMarketData* ZCMarket, PnlVec
    t* OptionPriceVect1, PnlVect* OptionPriceVect2, int index_
    last, int index_first)
{
    double a, b, sigma;

    double delta_t, sqrt_delta_t;

    double current_rate, current_x, x_middle;

    int i, h;
    int NumberNode, index;

    PnlVect* Probas;
    Probas = pnl_vect_create(3);

    ///********* Model parameters *********///
    a = (ModelParam->MeanReversion);
    b = (ModelParam->LongTermMean);
    sigma = (ModelParam->Volatility);

    delta_t = GET(Meth->t, 1) - GET(Meth->t,0); // = t[i] -
     t[i-1]
    sqrt_delta_t = sqrt(delta_t);

    for(i = index_last-1; i>=index_first; i--)
    {
        NumberNode = (int) ((GET(Meth->Xmax, i) - GET(Meth-
    >Xmin, i)) / (Meth->delta_x) + 0.1);

        pnl_vect_resize(OptionPriceVect1, NumberNode +1);
    // OptionPriceVect1 := Price of the bond in the tree at
    time t(i)

        // Loop over the node at the time i
        for(h = 0 ; h<= NumberNode ; h++)
        {
```

```
            current_x = x_value(i, h, Meth);
            current_rate = R(current_x, sigma) + GET(Meth->
    alpha,i);

            x_middle = MiddleNode(Meth, i, a, b, sigma,
    current_x, sqrt_delta_t, Probas);

            index = (int) ((x_middle-GET(Meth->Xmin,i+1))/(
    Meth->delta_x) + 0.1);

            LET(OptionPriceVect1,h) = exp(-current_rate*de
    lta_t) * ( GET(Probas,2) * GET(OptionPriceVect2, index+1) +
     GET(Probas,1) * GET(OptionPriceVect2, index) + GET(Prob
    as,0) * GET(OptionPriceVect2, index-1)); // Backward computa
    tion of the bond price
        }

        pnl_vect_clone(OptionPriceVect2, OptionPriceVect1);
     // Copy OptionPriceVect1 in OptionPriceVect2

    } // END of the loop on i (time)

    pnl_vect_free(&Probas);
}

/// Price at time "s" of a ZC bond maturing at "T" using a
    trinomial tree.
static double tr_cirpp1d_zcbond(TreeCIRpp1D* Meth, ModelCIR
    pp1D* ModelParam, ZCMarketData* ZCMarket, double T, double
    s, double r)
{
    double sigma;
    double delta_t, delta_r;
    double current_rate, current_x;

    double theta, OptionPrice1, OptionPrice2;
    double OptionPrice;
    int i_s;
    int j_r;

    PnlVect* Probas;
```

```
PnlVect* OptionPriceVect1; // Matrix of prices of the
option at i
PnlVect* OptionPriceVect2; // Matrix of prices of the
option at i+1

Probas = pnl_vect_create(3);
OptionPriceVect1 = pnl_vect_create(1);
OptionPriceVect2 = pnl_vect_create(1);

///********* Model parameters *********///
//a = (ModelParam->MeanReversion);
//b = (ModelParam->LongTermMean);
sigma = (ModelParam->Volatility);
current_x = ModelParam->Initialx0; // x(0)

delta_t = GET(Meth->t, 1) - GET(Meth->t,0); // = t[i] -
 t[i-1]

///****************** Computation of the vector of payo
ff at the maturity of the option ******************///
ZCBond_InitialPayoffCIRpp1D(Meth,OptionPriceVect2);

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

i_s = indiceTimeCIRpp1D(Meth, s); // Localisation of s
on the tree

if(i_s==0) // If s=0
{
    ZCBond_BackwardIterationCIRpp1D(Meth, ModelParam,
ZCMarket, OptionPriceVect1, OptionPriceVect2, Meth->Ngrid, 1
);

    current_rate = R(current_x, sigma) + GET(Meth->alp
ha,0);

    OptionPrice = exp(-current_rate*delta_t) * ( GET(
Probas,2) * GET(OptionPriceVect1, 2) + GET(Probas,1) * GET(
OptionPriceVect1,1) + GET(Probas,0) * GET(OptionPriceVect1, 0));
}
```

```
else
{   // We compute the price of the option as a linear
interpolation of the prices at the nodes r(i_s,j_r) and r(i_s,
j_r+1)

    j_r = (int) ((2 * sqrt(r-GET(Meth->alpha,i_s)) / si
gma - GET(Meth->Xmin,i_s)) / (Meth->delta_x) + 0.1); // r
between r(j_r) et r(j_r+1)

    if(j_r < 0 || j_r > (GET(Meth->Xmax,i_s)-GET(Meth->
Xmin,i_s))/(Meth->delta_x)-1)
    {
      printf("WARNING : Instantaneous futur spot rate
is out of tree{n");
      exit(EXIT_FAILURE);
    }

    ZCBond_BackwardIterationCIRpp1D(Meth, ModelParam,
ZCMarket, OptionPriceVect1, OptionPriceVect2, Meth->Ngrid,
i_s);

    current_x = x_value(i_s, j_r, Meth);

    current_rate = R(current_x, sigma) + GET(Meth->alp
ha,i_s);

    delta_r = R(x_value(i_s, j_r+1, Meth), sigma) -
current_x;

    theta = (r - current_rate)/ delta_r ;

    OptionPrice1 = GET(OptionPriceVect1, j_r);

    OptionPrice2 = GET(OptionPriceVect1, j_r + 1);

    OptionPrice = (1-theta) * OptionPrice1 + theta *
OptionPrice2 ;
}

pnl_vect_free(& OptionPriceVect1);
```

```
    pnl_vect_free(& OptionPriceVect2);
    pnl_vect_free(& Probas);

    return OptionPrice;

}


static int tr_zcbond1d(int flat_flag,double t,double r0,
    double a, double b, double sigma, double T,int N_steps,double *
    price)
{
  TreeCIRpp1D Tr;
  ModelCIRpp1D 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(T > GET(ZCMarket.tm,ZCMarket.Nvalue-1))
      {
          printf("{nError : time bigger than the last time
    value entered in initialyield.dat{n");
          exit(EXIT_FAILURE);
      }
  }

  ModelParams.MeanReversion = a;
  ModelParams.LongTermMean  = b;
  ModelParams.Volatility    = sigma;
```

```
  ModelParams.Initialx0     = 5;

  // Construction of the Time Grid
  SetTimegridCIRpp1D(&Tr, N_steps, t, T);

  // Construction of the tree, calibrated to the initial yi
    eld curve
  SetTreeCIRpp1D(&Tr, &ModelParams, &ZCMarket);

  //Price of Zero Coupon Bond
  *price = tr_cirpp1d_zcbond(&Tr, &ModelParams, &ZCMarket,
    T, t, r0);

  DeleteTreeCIRpp1D(&Tr);
  DeleteZCMarketData(&ZCMarket);

  return OK;
}



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

int CALC(TR_ZCBondCIRpp1D)(void *Opt,void *Mod,Pricing
    Method *Met)
{
  TYPEOPT* ptOpt=(TYPEOPT*)Opt;
  TYPEMOD* ptMod=(TYPEMOD*)Mod;


  return tr_zcbond1d( ptMod->flat_flag.Val.V_INT,
                      ptMod->T.Val.V_DATE,
                      MOD(GetYield)(ptMod),
                      ptMod->a.Val.V_DOUBLE,
                      ptMod->b.Val.V_DOUBLE,
                      ptMod->Sigma.Val.V_PDOUBLE,
                      ptOpt->BMaturity.Val.V_DATE,
                      Met->Par[0].Val.V_LONG,
                      &(Met->Res[0].Val.V_DOUBLE));
}
```

```
static int CHK_OPT(TR_ZCBondCIRpp1D)(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->Par[0].Val.V_LONG=50;
    }
  return OK;
}

PricingMethod MET(TR_ZCBondCIRpp1D)=
{
  "TR_Cirpp1d_ZCBond",
  {{"TimeStepNumber",LONG,{100},ALLOW},
   {" ",PREMIA_NULLTYPE,{0},FORBID}},
  CALC(TR_ZCBondCIRpp1D),
  {{"Price",DOUBLE,{100},FORBID},{" ",PREMIA_NULLTYPE,{0},
    FORBID}},
  CHK_OPT(TR_ZCBondCIRpp1D),
  CHK_ok,
  MET(Init)
} ;
```

References