

[Help](#)

```
#include "
href../../../../mod/blackkarasinski1d/blackkarasinski1d_std/blackkarasinski1d_std
#include "pnl/pnl_vector.h"
#include "
href../../../../common/math/InterestRateModelTree/TreeShortRate/TreeShortRate_h_src
#include "
href../../../../common/math/read_market_zc/InitialYieldCurve_h_src.pdfmath/read_mar

//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_ZCBondBK1D)(void *Opt, void *Mod)
{
    return NONACTIVE;
}
int CALC(TR_ZCBondBK1D)(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 Black&Ka
/// ZCMarketData       : structure that contains the Zero Coupon Bond prices of the ma

/// Computation of the payoff of the ZC bond at the final time of the tree (ie t
static void ZCBond_InitialPayoffBK1D(TreeShortRate *Meth, PnlVect *OptionPriceVe
{
    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.
static double tr_bk1d_zcbond(TreeShortRate *Meth, ModelParameters *ModelParam, Z
```

```

{
    int index_last, index_first;
    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_InitialPayoffBK1D(Meth, OptionPriceVect2);

    ///***** Backward computation of the bond price until time (s + d
    index_last = Meth->Ngrid;
    index_first = 0;

    BackwardIteration(Meth, ModelParam, OptionPriceVect1, OptionPriceVect2, index_

    OptionPrice = GET(OptionPriceVect1, 0);

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

    return OptionPrice;

} // FIN de la fonction ZCOption

static int tr_zcbond1d(int flat_flag, double r0, char *curve, double a, double s
{
    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 (T > 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;

SetTimeGrid(&Tr, N_steps, T);

SetTreeShortRate(&Tr, &ModelParams, &ZCMarket, &func_model_bk1d, &func_model_d

//Price of a ZC bond
*price = tr_bk1d_zcbond(&Tr, &ModelParams, &ZCMarket, T);

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

return OK;
}

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

int CALC(TR_ZCBondBK1D)(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->a.Val.V_DOUBLE,
        ptMod->Sigma.Val.V_PDOUBLE,
        ptOpt->BMaturity.Val.V_DATE - ptMod->T.Val.V_DATE,
        Met->Par[0].Val.V_LONG,
        &(Met->Res[0].Val.V_DOUBLE));
}
static int CHK_OPT(TR_ZCBondBK1D)(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_ZCBondBK1D) =
{
    "TR_BlackKarasinski1d_ZCBond",
    { {"TimeStepNumber", LONG, {100}, ALLOW},
      {" ", PREMIA_NULLTYPE, {0}, FORBID}
    },
    CALC(TR_ZCBondBK1D),
    {"Price", DOUBLE, {100}, FORBID}, {" ", PREMIA_NULLTYPE, {0}, FORBID}},
    CHK_OPT(TR_ZCBondBK1D),
    CHK_ok,
    MET(Init)
} ;

```

