

## [Help](#)

```
#include "
href../../mod/bharchiarella1d/bharchiarella1d_std/bharchiarella1d_std_h_src

/*****/
static double f0_cf(double t, double beta0, double beta1, double eta)
{
    return (beta0 + beta1 * (1 - exp(-eta * t)));
}
/*****/

/*static double f2(double t,double beta1,double eta)
{
    return( beta1*eta*exp(-eta*t));
}*/
/*****/
/*static double D(double t,double beta0,double beta1,double eta,double lambda)
{

    return(f2(t,beta1,eta)+lambda*f0_cf(t,beta0,beta1,eta));

}
*/
/*****/
static double alpha(double t, double tau_alpha, double lambda)
{
    return (exp(-lambda * t) / (exp(-lambda * tau_alpha) - exp(-lambda * t)));
}

/*****/
static double psi_cf(double t, double x, double y, double lambda, double tau, double eta)
{
    if (t > 0)
    {
        if (t < tau)
        {
            return (lambda * alpha(t, tau, lambda) * (x - f0_cf(t, beta0, beta1, eta) -
                (y - f0_cf(tau, beta0, beta1, eta))));
        }
    }
}
```

```

        else
        {
            return (0.0);
        }
    }
    else
    {
        return (0.0);
    }
}
/*****
/*static double mur(double t,double x,double y,double lambda,double beta0,double
{
    return(D(t,beta0,beta1,eta,lambda)+psi(t,x,y,lambda,tau,beta0,beta1,eta)-lambd
}
*/
/*****/

/*****/
static double beta_cf(double t, double T, double lambda)
{
    return (1 / lambda * (1 - exp(-lambda * (T - t))));
}
/*****/

static double bond_ratio(double t, double T, double beta0, double beta1, double
{
    return exp((beta0 + beta1) * (t - T) + beta1 * (exp(-eta * t) - exp(-eta * T))
}

/*****/

static int cf_bond1d(
    double t, /*
    double maturity_bond, /* maturité du zéro-coupon */
    /*
    double alpha0, /* Paramètres de la volatilité */
    double alphas, /*
    double alphaf,

    /*(t,T,r,f) = (alpha0+alphas*r+alphaf*f)^gamma*exp(-lambda(T-t)) */

```

```

double gamma0,      /*
double lambda,      /*
    */
double beta0,        /* Paramètres taux forward */
double beta1,        /* */
double eta,          /*  $f(0,t) = \beta_0 +$ 
                        *  $\beta_1(1-\exp(-\eta t))$ 
*/
double tau,
double *price)
{
    double x, y;

    /*Price*/
    double r00 = beta0; /* (r00,f00) */
    double f00 = beta0; /* à l'instant t */

    /*tau appears in forward rate volatility description*/
    /* constantes */
    if (tau > maturity_bond)
        return PREMIA_UNTREATED_TAU_BHAR_CHIARELLA;

    x = r00;
    y = f00;

    *price = bond_ratio(t, maturity_bond, beta0, beta1, eta) * exp(-beta_cf(t, mat

    return OK;
}

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

    return cf_bond1d(ptMod->T.Val.V_DATE, ptOpt->BMaturity.Val.V_DATE, ptMod->alph
}

```

```

static int CHK_OPT(CF_ZCBond)(void *Opt, void *Mod)
{
    if ((strcmp(((Option *)Opt)->Name, "ZeroCouponBond") == 0))
        return OK;
    else
        return WRONG;
}

static int MET(Init)(PricingMethod *Met, Option *Opt)
{
    if (Met->init == 0)
    {
        Met->init = 1;
    }
    return OK;
}

PricingMethod MET(CF_ZCBond) =
{
    "CF_BharChiarella1d_ZCBond",
    {
        {" ", PREMIA_NULLTYPE, {0}, FORBID}
    },
    CALC(CF_ZCBond),
    {"Price", DOUBLE, {100}, FORBID} , {" ", PREMIA_NULLTYPE, {0}, FORBID}},
    CHK_OPT(CF_ZCBond),
    CHK_ok,
    MET(Init)
} ;

```