Help

```
#include "
href../../../mod/cirpp1d/cirpp1d_stdi/cirpp1d_stdi_h_src.pdfcirpp1d_stdi.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2007+2) //The "#els
static int CHK_OPT(CF_ZCCallBondEuro)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(CF_ZCCallBondEuro)(void *Opt, void *Mod, PricingMethod *Met)
{
  return AVAILABLE_IN_FULL_PREMIA;
}
#else

/*Shift function of the CIR++ model*/
static double shift(double a, double b, double sigma, double f0_s, double s)
{
  /* the shift rate of the cir++ model for x(0)=0 */
  double  c;

  c = sqrt(a * a + 2 * sigma * sigma);

  return (f0_s - 2 * a * b * (exp(s * c) - 1) / (2 * c + (a + c) * (exp(s * c) -
}

static double A(double time, double a, double b, double sigma)
{
  double h = sqrt(SQR(a) + 2.*SQR(sigma));
  return pow(h * exp(0.5 * (a + h) * (time)) / (h + 0.5 * (a + h) * (exp(h * (ti
}

static double B(double time, double a, double b, double sigma)
{
  double h = sqrt(SQR(a) + 2.*SQR(sigma));
  return (exp(h * (time)) - 1.) / (h + 0.5 * (a + h) * (exp(h * (time)) - 1.));
}

/*Zero Coupon Bond*/
static double zcbond(double rcc, double a, double b, double sigma, double t, dou
```

```
{
  if (t == 0)
    {
      return BondPrice(T, ZCMarket);
    }
  else
    {
      double h, A, B, At, AT, shift, c;
      double f0_t, P0_t, P0_T, P0_t_plus, P0_t_minus;

      P0_t = BondPrice(t, ZCMarket);
      P0_T = BondPrice(T, ZCMarket);

      /*Computation of Forward rate*/
      P0_t_plus = BondPrice(t * (1. + INC), ZCMarket);
      P0_t_minus = BondPrice(t * (1. - INC), ZCMarket);
      f0_t = -(log(P0_t_plus) - log(P0_t_minus)) / (2.*t * INC);

      /*A,B coefficient*/
      h = sqrt(SQR(a) + 2.*SQR(sigma));
      B = 2.*(exp(h * (T - t)) - 1.) / (2.*h + (a + h) * (exp(h * (T - t)) - 1.)
      A = pow(h * exp(0.5 * (a + h) * (T - t)) / (h + 0.5 * (a + h) * (exp(h * (
      At = pow(h * exp(0.5 * (a + h) * (t)) / (h + 0.5 * (a + h) * (exp(h * (t))
      AT = pow(h * exp(0.5 * (a + h) * (T)) / (h + 0.5 * (a + h) * (exp(h * (T))

      c = sqrt(a * a + 2 * sigma * sigma);

      shift = (f0_t - 2 * a * b * (exp(t * c) - 1) / (2 * c + (a + c) * (exp(t *

      A = A * (P0_T * At) / (AT * P0_t) * exp(B * shift);

      /*Price*/
      return A * exp(-B * rcc);
    }
}

/*Call Option*/
static int zbc_cirpp1d(int flat_flag, double a, double b, double t, double sigma
{
  double K;
  double PtS, PtT, ATS, BTS;
```

```
double f0_t;
double p1, p2, p3, k1, k2, k3, psi, phi, rb;
double h;

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 = rcc;
  }

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);
      }
  }

/*Computation of Forward rate*/
h = sqrt(SQR(a) + 2.*SQR(sigma));

if (t - 0.5 * INC > 0)
  {
    f0_t = (log(BondPrice(t - 0.5 * INC, &ZCMarket)) - log(BondPrice(t + 0.5 *
  }
else
  {
    f0_t = -log(BondPrice(INC, &ZCMarket)) / INC;
  }
K = p->Par[0].Val.V_DOUBLE;

PtT = zcbond(rcc, a, b, sigma, t, T, &ZCMarket);
```

```
  PtS = zcbond(rcc, a, b, sigma, t, S, &ZCMarket);

  BTS = B(S - T, a, b, sigma);
  ATS = A(S - T, a, b, sigma);

  /*X^2 parameters*/
  rb = (log(ATS / K) + log(A(T, a, b, sigma) * BondPrice(S, &ZCMarket)) - log(A(
  if (rb < 0)
    {
      rb = 0;
    }
  phi = 2.*h / (SQR(sigma) * (exp(h * (T - t)) - 1.));
  psi = (a + h) / SQR(sigma);

  p1 = 2.*rb * (phi + psi + BTS);
  p2 = 4.*a * b / SQR(sigma);
  p3 = 2.*SQR(phi) * (rcc - shift(a, b, sigma, f0_t, t)) * exp(h * (T - t)) / (p

  k1 = 2.*rb * (phi + psi);
  k2 = p2;
  k3 = 2.*SQR(phi) * (rcc - shift(a, b, sigma, f0_t, t)) * exp(h * (T - t)) / (p

  /*Price of Call*/
  *price = PtS * pnl_cdfchi2n(p1, p2, p3) - K * PtT * pnl_cdfchi2n(k1, k2, k3);

  *delta = pnl_cdfchi2n(p1, p2, p3);

  return OK;
}


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

  return zbc_cirpp1d(ptMod->flat_flag.Val.V_INT, ptMod->a.Val.V_DOUBLE, ptMod->b
                     ptMod->Sigma.Val.V_PDOUBLE, MOD(GetYield)(ptMod),
                     MOD(GetCurve)(ptMod), ptOpt->BMaturity.Val.V_DATE,
                     ptOpt->OMaturity.Val.V_DATE, ptOpt->PayOff.Val.V_NUMFUNC_1,
                     &(Met->Res[1].Val.V_DOUBLE));
```

```
}
static int CHK_OPT(CF_ZCCallBondEuro)(void *Opt, void *Mod)
{
  return strcmp(((Option *)Opt)->Name, "ZeroCouponCallBondEuro");
}
#endif //PremiaCurrentVersion


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

  return OK;
}

PricingMethod MET(CF_ZCCallBondEuro) =
{
  "CF_Cirpp1d_ZBCallEuro",
  {{" ", PREMIA_NULLTYPE, {0}, FORBID}},
  CALC(CF_ZCCallBondEuro),
  {{"Price", DOUBLE, {100}, FORBID}, {"Delta", DOUBLE, {100}, FORBID} , {" ", PR
  CHK_OPT(CF_ZCCallBondEuro),
  CHK_ok,
  MET(Init)
} ;
```