

[Help](#)

```

#include "cirpp1d_std.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
    (2007+2) //The "#else" part of the code will be freely av
    ailable after the (year of creation of this file + 2)
static int CHK_OPT(CF_Cap)(void *Opt, void *Mod)
{
    return NONACTIVE;
}
int CALC(CF_Cap)(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)-1)
        ));
}

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*(
        time))-1.)),2.*a*b/SQR(sigma));
}

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.)

```

```

    );
}

static double zcbond(double rcc,double a,double b,double sigma,double t,double T, ZCMarketData* ZCMarket)
{
    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*(T-t))-1.)), 2.*a*b/SQR(sigma));
        At=pow(h*exp(0.5*(a+h)*(t))/(h+0.5*(a+h)*(exp(h*(t))-1.)), 2.*a*b/SQR(sigma));
        AT=pow(h*exp(0.5*(a+h)*(T))/(h+0.5*(a+h)*(exp(h*(T))-1.)), 2.*a*b/SQR(sigma));

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

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

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

```

```

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

/*Call Option on Zero Coupon Bond*/
static double zbput(double a, double b, double sigma,
    double rcc, double t, double T, double S, double K, ZCMarketData*
    ZCMarket)
{
    double PtS, PtT, ATS, BTS;
    double f0_t;
    double p1, p2, p3, k1, k2, k3, psi, phi, rb;
    double h;

    /*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, ZCMar
ket))-log( BondPrice(t+0.5*INC, ZCMarket)))/INC;}
    else {f0_t = -log( BondPrice(INC, ZCMarket))/INC; }

    PtT=zbond(rcc,a,b,sigma,t,T, ZCMarket);
    PtS=zbond(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, ZCMarke
t))-log(A(S,a,b,sigma)*BondPrice(T, ZCMarket)))/BTS;
    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))/(phi+psi+BTS);

    k1=2.*rb*(phi+psi);

```

```

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

    /*Price of Put by Parity*/

    return PtS*pnl_cdfchi2n(p1,p2,p3)-K*PtT*pnl_cdfchi2n(k1
    ,k2,k3) -PtS+K*PtT;;
}

/*Cap*/
static int cap_cirpp1d(int flat_flag,double a,double b,
    double date,double sigma,double rcc,double Nominal,double K,
    double periodicity,double first_payment,double contract_maturit
    y,double *price/*,double *delta*/)
{
    double sum,tim,tip;
    int i, nb_payment;

    ZCMarketData ZCMarket;

    /* Flag to decide to read or not ZC bond datas in "ini
    tialyields.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;
        ReadMarketData(&ZCMarket);
    }

    nb_payment = (int) ((contract_maturity-first_payment)
    /periodicity + 0.1);

    /*Cap=Portfolio of zero-bond Put options*/
    sum=0.;

```

```

    for(i=0;i<nb_payement;i++)
    {
        tim=first_payement+(double)i*periodicity;
        tip=tim+periodicity;
        sum+=(1.+K*periodicity)*zbput(a,b,sigma,rcc,date,
        tim,tip,1./(1.+K*periodicity), &ZCMarket);
    }

    /*Price*/
    *price=Nominal*sum;
    /*Delta*/
    /**delta=0.;*/
    return OK;
}

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

    return cap_cirpp1d(ptMod->flat_flag.Val.V_INT,
                        ptMod->a.Val.V_DOUBLE,
                        ptMod->b.Val.V_DOUBLE,
                        ptMod->T.Val.V_DATE,
                        ptMod->Sigma.Val.V_PDOUBLE,
                        MOD(GetYield)(ptMod),
                        ptOpt->Nominal.Val.V_PDOUBLE,
                        ptOpt->FixedRate.Val.V_PDOUBLE,
                        ptOpt->ResetPeriod.Val.V_DATE,
                        ptOpt->FirstResetDate.Val.V_DATE,
                        ptOpt->BMaturity.Val.V_DATE,
                        &(Met->Res[0].Val.V_DOUBLE));
}

static int CHK_OPT(CF_Cap)(void *Opt, void *Mod)
{
    return strcmp( ((Option*)Opt)->Name,"Cap");
}

```

```

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

    return OK;
}

PricingMethod MET(CF_Cap)=
{
    "CF_Cirpp1d_Cap",
    {{" ",PREMIA_NULLTYPE,{0},FORBID}},
    CALC(CF_Cap),
    {{"Price",DOUBLE,{100},FORBID}/*,{"Delta",DOUBLE,{100},FORBID}
    /*,{" ",PREMIA_NULLTYPE,{0},FORBID}},
    CHK_OPT(CF_Cap),
    CHK_ok,
    MET(Init)
} ;

```

References