

## Help

```

#include "libor_affine_cir1d_std.h"
#include "math/libor_affine_model/libor_affine_framework.h"
#include "math/libor_affine_model/libor_affine_pricing.h"
#include "math/libor_affine_model/libor_affine_models.h"
#include "pnl/pnl_cdf.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2011+2) //The "#els
static int CHK_OPT(CF_LibAffCir1d_Direct_CapFloor)(void *Opt, void *Mod)
{
    return NONACTIVE;
}
int CALC(CF_LibAffCir1d_Direct_CapFloor)(void *Opt, void *Mod, PricingMethod *Me
{
    return AVAILABLE_IN_FULL_PREMIA;
}
#else

static double cf_caplet_direct(StructLiborAffine *LiborAffine, int caplet_floorl
{
    double x0, lambda, theta, eta, SQR_eta;
    double TN = LET(LiborAffine->TimeDates, (LiborAffine->TimeDates)->size - 1), c
    double P0_Tk1, P0_Tk2, K_tilde, log_K_tilde, K_star;
    double phi_k1, phi_k2, psi_k1, psi_k2;
    double P1 = 0., Q1 = 0., P2 = 0., Q2 = 0., x, df, pnonc, bound;
    double Ak, Bk, a_Tk1, b_Tk1, dzeta_1, sigma_1, dzeta_2, sigma_2;
    int k, which = 1, status;
    dcomplex uk1, uk2, phi, psi;

    x0      = GET(LiborAffine->ModelParams, 0);
    lambda  = GET(LiborAffine->ModelParams, 1);
    theta   = GET(LiborAffine->ModelParams, 2);
    eta     = GET(LiborAffine->ModelParams, 3);
    SQR_eta = SQR(eta);

    P0_Tk1 = BondPrice(Tk1, LiborAffine->ZCMarket);
    P0_Tk2 = BondPrice(Tk2, LiborAffine->ZCMarket);

    K_tilde = (1 + cap_period * cap_strike);
    log_K_tilde = log(K_tilde);

```

```

K_star = P0_Tk2 * K_tilde;

k = indiceTimeLiborAffine(LiborAffine, Tk1);
uk1 = Complex(GET(LiborAffine->MartingaleParams, k), 0.);
uk2 = Complex(GET(LiborAffine->MartingaleParams, k + 1), 0.);

phi_psi_t_v(TN - Tk1, uk1, LiborAffine, &phi, &psi);
phi_k1 = Creal(phi);
psi_k1 = Creal(psi);

phi_psi_t_v(TN - Tk1, uk2, LiborAffine, &phi, &psi);
phi_k2 = Creal(phi);
psi_k2 = Creal(psi);

Ak = phi_k1 - phi_k2;
Bk = psi_k1 - psi_k2;

a_Tk1 = exp(-lambda * Tk1);
if (lambda == 0.) b_Tk1 = Tk1;
else b_Tk1 = (1. - a_Tk1) / lambda;

dzeta_1 = 1 - 2 * SQR_eta * b_Tk1 * psi_k1;
sigma_1 = Bk * SQR_eta * b_Tk1 / dzeta_1;
x = (log_K_tilde - Ak) / sigma_1;
if (x < 0)
{
    P1 = 0.;
    Q1 = 1.;
}
else
{
    df = lambda * theta / SQR_eta;
    pnonc = x0 * a_Tk1 / (SQR_eta * b_Tk1 * dzeta_1);
    pnl_cdf_chn(&which, &P1, &Q1, &x, &df, &pnonc, &status, &bound);
}

dzeta_2 = 1 - 2 * SQR_eta * b_Tk1 * psi_k2;
sigma_2 = Bk * SQR_eta * b_Tk1 / dzeta_2;
x = (log_K_tilde - Ak) / sigma_2;
if (x < 0)
{

```

```

        P2 = 0.;
        Q2 = 1.;
    }
    else
    {
        pnonc = x0 * a_Tk1 / (SQR_eta * b_Tk1 * dzeta_2);
        pnl_cdf_chn(&which, &P2, &Q2, &x, &df, &pnonc, &status, &bound);
    }

    if (caplet_floorlet == 0) price = P0_Tk1 * Q1 - K_star * Q2;
    else price = -P0_Tk1 * P1 + K_star * P2;

    return price;
}

static int cf_capfloor_direct_libaff_cir1d(int InitYieldCurve_flag, double R_flat)
{
    int i, nb_payment = pnl_iround((cap_end - cap_start) / cap_period);
    double caplet_Tk1, caplet_Tk2, caplet_price;
    StructLiborAffine LiborAffine;
    ZCMarketData ZCMarket;
    PnlVect *ModelParams = pnl_vect_create(4);

    LET(ModelParams, 0) = x0;
    LET(ModelParams, 1) = lambda;
    LET(ModelParams, 2) = theta;
    LET(ModelParams, 3) = eta;

    ZCMarket.filename = curve;
    SetInitYieldCurve(InitYieldCurve_flag, R_flat, &ZCMarket);

    CreateStructLiborAffine(&LiborAffine, &ZCMarket, cap_start, cap_end, cap_period);

    caplet_Tk1 = cap_start;
    caplet_Tk2 = caplet_Tk1 + cap_period;
    *cap_price = 0.;
    for (i = 0; i < nb_payment; i++)
    {
        caplet_price = cap_nominal * cf_caplet_direct(&LiborAffine, cap_floor, cap_end, caplet_Tk1, caplet_Tk2);
        *cap_price += caplet_price;
    }
}

```

```

        caplet_Tk1 += cap_period;
        caplet_Tk2 += cap_period;
    }

    FreeStructLiborAffine(&LiborAffine);

    return OK;
}

//***** PREMIA FUNCTIONS *****
int CALC(CF_LibAffCir1d_Direct_CapFloor)(void *Opt, void *Mod, PricingMethod *Me
{
    TYPEOPT *ptOpt = (TYPEOPT *)Opt;
    TYPEMOD *ptMod = (TYPEMOD *)Mod;

    int cap_floor = (((ptOpt->PayOff.Val.V_NUMFUNC_1)->Compute) == &Call);

    return  cf_capfloor_direct_libaff_cir1d(ptMod->flat_flag.Val.V_INT,
                                             MOD(GetYield)(ptMod),
                                             MOD(GetCurve)(ptMod),
                                             ptMod->x0.Val.V_DOUBLE,
                                             ptMod->lambda.Val.V_PDOUBLE,
                                             ptMod->theta.Val.V_DOUBLE,
                                             ptMod->eta.Val.V_PDOUBLE,
                                             ptOpt->FirstResetDate.Val.V_DATE - ptM
                                             ptOpt->BMaturity.Val.V_DATE - ptMod->T
                                             ptOpt->ResetPeriod.Val.V_DATE,
                                             ptOpt->FixedRate.Val.V_PDOUBLE,
                                             ptOpt->Nominal.Val.V_PDOUBLE,
                                             cap_floor,
                                             &(Met->Res[0].Val.V_DOUBLE));
}

static int CHK_OPT(CF_LibAffCir1d_Direct_CapFloor)(void *Opt, void *Mod)
{
    if ((strcmp(((Option *)Opt)->Name, "Cap") == 0) || (strcmp(((Option *)Opt)->Na
        return OK;
    else
        return WRONG;
}

```

```
#endif //PremiaCurrentVersion
```

```
static int MET(Init)(PricingMethod *Met, Option *Opt)
{
    if (Met->init == 0)
    {
        Met->init = 1;
        Met->HelpFilenameHint = "cf_libor_affine_cir1d_capfloor_direct";
    }
    return OK;
}
```

```
PricingMethod MET(CF_LibAffCir1d_Direct_CapFloor) =
{
    "CF_LibAffCir1d_Direct_CapFloor",
    {{" ", PREMIA_NULLTYPE, {0}, FORBID}},
    CALC(CF_LibAffCir1d_Direct_CapFloor),
    {{"Price", DOUBLE, {100}, FORBID}, {" ", PREMIA_NULLTYPE, {0}, FORBID}},
    CHK_OPT(CF_LibAffCir1d_Direct_CapFloor),
    CHK_ok,
    MET(Init)
} ;
```