

[Help](#)

```
#include "doublehes1d_std.h"
#include "pnl/pnl_integration.h"
#include "pnl/pnl_finance.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2013+2) //The "#els
static int CHK_OPT(CF_Carr_DoubleHeston)(void *Opt, void *Mod)
{
    return NONACTIVE;
}
int CALC(CF_Carr_DoubleHeston)(void *Opt, void *Mod, PricingMethod *Met)
{
    return AVAILABLE_IN_FULL_PREMIA;
}
#else

static dcomplex d_heston(double b, double rho, double sigma, dcomplex k)
{
    dcomplex b1, b2, b3;
    b1.r = b + rho * sigma * k.i;
    b1.i = -rho * sigma * k.r;
    b2 = Cmul(b1, b1);
    b1.r = sigma * sigma * k.r;
    b1.i = sigma * sigma * k.i;
    b3.r = k.r;
    b3.i = k.i + 1.;
    return Csqrt(Cadd(b2, Cmul(b1, b3)));
}

static dcomplex G_heston(double b, double rho, double sigma, dcomplex k, dcomplex d)
{
    dcomplex b1;
    b1.r = b + rho * sigma * k.i;
    b1.i = -rho * sigma * k.r;
    return Cdiv(Csub(b1, d), Cadd(b1, d));
}

static void A_B_heston(double b, double theta, double rho, double sigma, dcomplex k, dcomplex d)
{

```

```

dcomplex G, d, b1;

d = d_heston(b, rho, sigma, k);
G = G_heston(b, rho, sigma, k, d);
b1.r = b + rho * sigma * k.i;
b1.i = -rho * sigma * k.r;
*A = RCmul(b * theta / (sigma * sigma), Csub(RCmul(t, Csub(b1, d)), RCmul(2.,

*B = CMul(RCmul(1. / (sigma * sigma), Csub(b1, d)), Cdiv(RCsub(1., Cexp(RCmul(
return;
}

static dcomplex heston_psi(dcomplex k, double t, double x, double V01, double V02)
{
dcomplex A1, A2, B1, B2, c;
A_B_heston(b1, theta1, rho1, sigma1, k, t, &A1, &B1);
A_B_heston(b2, theta2, rho2, sigma2, k, t, &A2, &B2);
c.r = -k.i * x;
c.i = k.r * x;
return Cexp(Cadd(Cadd(Cadd(Cadd(c, A1), A2), RCmul(V01, B1)), RCmul(V02, B2)))
}

static dcomplex bs_psi(dcomplex k, double t, double x, double V01, double V02, dcomplex d)
{
dcomplex c, d;
double a, r1, r2;
r1 = (1. - exp(-b1 * t)) * 0.5 / b1;
r2 = (1. - exp(-b2 * t)) * 0.5 / b2;
a = theta1 * r1 + theta2 * r2 - 0.5 * (theta1 + theta2) * t;
c.r = k.r;
c.i = k.i + 1.;
d.r = -k.i * x;
d.i = k.r * x;
return Cexp(Cadd(d, RCmul(a - r1 * V01 - r2 * V02, CMul(k, c))));
}

static double re_eiuk_psi_m_tildepsi_bis(double u, double s0, double K, double T)
{
dcomplex k, psi, tildepsi;
double logK, x;
x = (r - divid) * T + log(s0);

```

```

    logK = log(K);
    k.r = u;
    k.i = -0.5;
    psi = heston_psi(k, T, x, V01, V02, b1, b2, theta1, theta2, sigma1, sigma2, rho1, rho2, rho3);
    tildepsi = bs_psi(k, T, x, V01, V02, b1, b2, theta1, theta2, sigma1, sigma2, rho1, rho2, rho3);
    return exp(-r * T + 0.5 * logK) * M_1_PI * (cos(u * logK) * (Creal(psi) - Creal(tildepsi)));
}

```

```

static double lafunction(double u, void *ptr)
{
    PnlVect *ptrbis = ptr;
    double s0, K, T, r, divid, V01, V02, b1, b2, theta1, theta2, sigma1, sigma2, rho1, rho2, rho3;
    s0 = pnl_vect_get(ptrbis, 0);
    K = pnl_vect_get(ptrbis, 1);
    T = pnl_vect_get(ptrbis, 2);
    r = pnl_vect_get(ptrbis, 3);
    divid = pnl_vect_get(ptrbis, 4);
    V01 = pnl_vect_get(ptrbis, 5);
    V02 = pnl_vect_get(ptrbis, 6);
    b1 = pnl_vect_get(ptrbis, 7);
    b2 = pnl_vect_get(ptrbis, 8);
    theta1 = pnl_vect_get(ptrbis, 9);
    theta2 = pnl_vect_get(ptrbis, 10);
    sigma1 = pnl_vect_get(ptrbis, 11);
    sigma2 = pnl_vect_get(ptrbis, 12);
    rho1 = pnl_vect_get(ptrbis, 13);
    rho2 = pnl_vect_get(ptrbis, 14);
    rho3 = pnl_vect_get(ptrbis, 15);

    chu = -log(u);
    du = 1. / u;
    return re_eiuk_psi_m_tildepsi_bis(chu, s0, K, T, r, divid, V01, V02, b1, b2, theta1, theta2, sigma1, sigma2, rho1, rho2, rho3, du);
}

```

```

int CFDoubleHeston(double s0, NumFunc_1 *p, double T, double r, double divid, double d)
{
    int flag_call;
    double K;
    double erreurabs, callbs, primeheston;
    int neval;
}

```

```

PnlVect *ptr = NULL;
PnlFunc F;
double rho3 = 0.;

K = p->Par[0].Val.V_PDOUBLE;
if ((p->Compute) == &Call)
    flag_call = 1;
else
    flag_call = 0;

ptr = pnl_vect_create(16);
pnl_vect_set(ptr, 0, s0);
pnl_vect_set(ptr, 1, K);
pnl_vect_set(ptr, 2, T);
pnl_vect_set(ptr, 3, r);
pnl_vect_set(ptr, 4, divid);
pnl_vect_set(ptr, 5, V01);
pnl_vect_set(ptr, 6, V02);
pnl_vect_set(ptr, 7, b1);
pnl_vect_set(ptr, 8, b2);
pnl_vect_set(ptr, 9, theta1);
pnl_vect_set(ptr, 10, theta2);
pnl_vect_set(ptr, 11, sigma1);
pnl_vect_set(ptr, 12, sigma2);
pnl_vect_set(ptr, 13, rho1);
pnl_vect_set(ptr, 14, rho2);
pnl_vect_set(ptr, 15, rho3);

F.F = lafunction;
F.params = ptr;
pnl_cf_call_bs(s0, K, T, r, divid, sqrt(theta1 + theta2 + (V01 - theta1) * (1.

pnl_integration_GK(&F, 0., 1., 0.0001, 0.000025, &primeheston, &erreurabs, &ne
//Call Price
*ptprice = callbs + primeheston;
//Put Price
if (flag_call == 0)
    *ptprice = *ptprice + K * exp(-r * T) - s0 * exp(-divid * T);

pnl_vect_free(&ptr);

```

```

    return OK;
}
int CALC(CF_Carr_DoubleHeston)(void *Opt, void *Mod, PricingMethod *Met)
{
    TYPEOPT *ptOpt = (TYPEOPT *)Opt;

    TYPEMOD *ptMod = (TYPEMOD *)Mod;
    double r, divid;

    if (ptMod->Sigma.Val.V_PDOUBLE == 0.0)
    {
        Fprintf(TOSCREEN, "BLACK-SCHOLES MODEL\ n\ n\ n");
        return WRONG;
    }
    else
    {
        r = log(1. + ptMod->R.Val.V_DOUBLE / 100.);
        divid = log(1. + ptMod->Divid.Val.V_DOUBLE / 100.);

        return CFDDoubleHeston(ptMod->S0.Val.V_PDOUBLE,
                                ptOpt->PayOff.Val.V_NUMFUNC_1,
                                ptOpt->Maturity.Val.V_DATE - ptMod->T.Val.V_DATE,
                                r, divid, ptMod->Sigma0.Val.V_PDOUBLE,
                                ptMod->Sigma0V.Val.V_PDOUBLE,
                                ptMod->LongRunVariance.Val.V_PDOUBLE,
                                ptMod->LongRunVarianceV.Val.V_PDOUBLE,
                                ptMod->MeanReversion.Val.V_PDOUBLE,
                                ptMod->MeanReversionV.Val.V_PDOUBLE,
                                ptMod->Sigma.Val.V_PDOUBLE,
                                ptMod->SigmaV.Val.V_PDOUBLE,
                                ptMod->Rho.Val.V_DOUBLE,
                                ptMod->RhoSV2.Val.V_DOUBLE,
                                &(Met->Res[0].Val.V_DOUBLE)
                                );
    }
}

static int CHK_OPT(CF_Carr_DoubleHeston)(void *Opt, void *Mod)
{
    if ((strcmp(((Option *)Opt)->Name, "CallEuro") == 0) || (strcmp(((Option *)Opt)

```

```
        return OK;

    return  WRONG;
}
#endif //PremiaCurrentVersion

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

    return OK;
}

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