

Help

```
#include "hes1d_std.h"
#include "pnl/pnl_finance.h"
#include "pnl/pnl_mathtools.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2010+2) //The "#els
static int CHK_OPT(AP_BGM_Heston)(void *Opt, void *Mod)
{
    return NONACTIVE;
}
int CALC(AP_BGM_Heston)(void *Opt, void *Mod, PricingMethod *Met)
{
    return AVAILABLE_IN_FULL_PREMIA;
}
#else
////////////////////////////////////

/*****
Computation of the partial derivatives given by formula (2.13) page 7
*****/

static int greeksBS(double x, double y, double K, double T, double r, double divid

                double *Pxy, double *Pyy, double *Pxxy, double *Pxxyy)
{

    double f, g, fg;

    f = (log(K) - x - r * T + divid * T) / sqrt(y) + 0.5 * sqrt(y);
    g = f - sqrt(y);
    fg = f * g;

    *Pxy = (0.5 / (sqrt(2 * M_PI) * y * sqrt(y))) * (exp(x) * exp(-divid * T) * (s

    *Pyy = (0.25 / (sqrt(2 * M_PI) * SQR(y))) * (exp(x) * exp(-divid * T) * (-2 *

    *Pxxy = (0.5 / (sqrt(2 * M_PI) * y * sqrt(y))) * (exp(x) * exp(-divid * T) * (

    *Pxxyy = (0.25 / (sqrt(2 * M_PI) * CUB(y))) *
```

```

        (exp(x) * exp(-divid * T) * ((sqrt(y) - g) * ((-2 * f - g + SQR(f) *
        * exp(-0.5 * SQR(g)) - K * exp(-r * T) * (9 * f + 6 * g - 3 * f * SQ

return 0;

}

static int ApBGMHeston(double S, NumFunc_1 *p, double T, double r, double divid
        double v0, double kappa, double theta, double sigma, doub
        double incr, double *ptprice, double *ptdelta)
{
    double K, m0, m1, p0, p1, q0, q1, r0, r1;
    double var, a1, a2, b0, b2;
    double kappaT;
    double Pxy, Pyy, Pxxy, Pxyy;
    double Pxyhu, Pyyhu, Pxxyhu, Pxyyhu, Pxyhd, Pyyhd, Pxxyhd, Pxyyhd;
    double BS_price, BS_delta;

    kappaT = kappa * T;
    K = p->Par[0].Val.V_PDDOUBLE;

    /*****
    * Explicit computations fo constant parameter case see page 7
    *****/

    m0 = exp(-kappaT) * (-1. + exp(kappaT)) / kappa;
    m1 = T - m0;
    p0 = exp(-kappaT) * (-1. + exp(kappaT) - kappaT) / (SQR(kappa));
    p1 = exp(-kappaT) * (2. + exp(kappaT) * (kappaT - 2.) + kappaT) / (SQR(kappa));
    q0 = exp(-kappaT) * (-kappaT * (kappaT + 2.) + 2.*exp(kappaT) - 2.) / (2.*CUB(
    q1 = exp(-kappaT) * (2.*exp(kappaT) * (kappaT - 3.) + kappaT * (kappaT + 4.) +
    r0 = exp(-2.*kappaT) * (-4.*exp(kappaT) * kappaT + 2.*exp(2.*kappaT) - 2.) / (
    r1 = exp(-2.*kappaT) * (4.*exp(kappaT) * (kappaT + 1.) + exp(2.*kappaT) * (2.*

    var = m0 * v0 + m1 * theta;
    a1 = rho * sigma * (p0 * v0 + p1 * theta);
    a2 = (rho * sigma) * (rho * sigma) * (q0 * v0 + q1 * theta);
    b0 = sigma * sigma * (r0 * v0 + r1 * theta);

```

```

b2 = 0.5 * a1 * a1;

greeksBS(log(S), var, K, T, r, divid, &Pxy, &Pyy, &Pxyy, &Pxyyy);
greeksBS(log(S * (1. + incr)), var, K, T, r, divid, &Pxyhu, &Pyyhu, &Pxyyhu, &Pxyyyhu);
greeksBS(log(S * (1. - incr)), var, K, T, r, divid, &Pxyhd, &Pyyhd, &Pxyyhd, &Pxyyyhd);

/* Price given by formula (2.13) page 7*/
if ((p->Compute) == &Put)
{
    pnl_cf_put_bs(S, K, T, r, divid, sqrt(var / T), &BS_price, &BS_delta);
    *ptprice = BS_price + a1 * Pxy + a2 * Pxyy + b0 * Pyy + b2 * Pxyyy;
    *ptdelta = BS_delta + 0.5 * (a1 * (Pxyhu - Pxyhd) + a2 * (Pxyyhu - Pxyyhd));
} // Call case
else
{
    pnl_cf_call_bs(S, K, T, r, divid, sqrt(var / T), &BS_price, &BS_delta);
    *ptprice = BS_price + a1 * Pxy + a2 * Pxyy + b0 * Pyy + b2 * Pxyyy;
    *ptdelta = BS_delta + 0.5 * (a1 * (Pxyhu - Pxyhd) + a2 * (Pxyyhu - Pxyyhd));
}

return OK;
}

int CALC(AP_BGM_Heston)(void *Opt, void *Mod, PricingMethod *Met)
{
    TYPEOPT *ptOpt = (TYPEOPT *)Opt;
    TYPEMOD *ptMod = (TYPEMOD *)Mod;
    double r, divid;

    if (ptMod->Sigma.Val.V_PDDOUBLE == 0.0)
    {
        Fprintf(TOSCREEN, "BLACK-SHOLES 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 ApBGMHeston(ptMod->S0.Val.V_PDDOUBLE,

```

```

        ptOpt->PayOff.Val.V_NUMFUNC_1,
        ptOpt->Maturity.Val.V_DATE - ptMod->T.Val.V_DATE,
        r,
        divid, ptMod->Sigma0.Val.V_PDOUBLE,
        ptMod->MeanReversion.Val.V_PDOUBLE,
        ptMod->LongRunVariance.Val.V_PDOUBLE,
        ptMod->Sigma.Val.V_PDOUBLE,
        ptMod->Rho.Val.V_PDOUBLE,
        Met->Par[0].Val.V_DOUBLE,
        &(Met->Res[0].Val.V_DOUBLE),
        &(Met->Res[1].Val.V_DOUBLE)
    );
}

}

static int CHK_OPT(AP_BGM_Heston)(void *Opt, void *Mod)
{
    if ((strcmp(((Option *)Opt)->Name, "CallEuro") == 0)
        || (strcmp(((Option *)Opt)->Name, "PutEuro") == 0))
        return OK;
    return WRONG;
}

#endif //PremiaCurrentVersion
static int MET(Init)(PricingMethod *Met, Option *Opt)
{
    if (Met->init == 0)
    {
        Met->init = 1;
        Met->Par[0].Val.V_DOUBLE = 0.01;
    }
    return OK;
}

PricingMethod MET(AP_BGM_Heston) =
{
    "AP_BGM_Heston",
    { {"Delta increment", DOUBLE, {100}, ALLOW},
      {" ", PREMIA_NULLTYPE, {0}, FORBID}
    },

```

```
CALC(AP_BGM_Heston),
{ {"Price", DOUBLE, {100}, FORBID},
  {"Delta", DOUBLE, {100}, FORBID} ,
  {" ", PREMIA_NULLTYPE, {0}, FORBID}
},
CHK_OPT(AP_BGM_Heston),
CHK_ok,
MET(Init)
};
```