

Help

```
#include "timehes1d_std.h"
#include "pnl/pnl_cdf.h"
#include "pnl/pnl_finance.h"
#include "pnl/pnl_root.h"

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

int  expansion_terms(double kappa, double v0, double theta, double t,
                    double T, double u, double *f0, double *f1,
                    double *f2, double *g1, double *g2, double *h1,
                    double *h2, double *wu, double *w0u, double *wuu)
{
    double kappaT, kappat;

    kappaT = kappa * T;
    kappat = kappa * t;

    *f0 = exp(-2 * kappaT) * (exp(2 * kappat) * (theta - 2 * v0) + exp(2 * kappaT)

    *f1 = exp(-kappaT) * (exp(kappaT) * ((-kappat + kappaT - 2) * theta + v0) - ex

    *f2 = exp(-kappa * (t + 3 * T)) * (2 * exp(kappa * (t + 3 * T)) * ((kappa * (T

    *g1 = (2 * exp(kappaT) * theta + exp(kappat) * (SQR(kappa) * SQR(t - T) * v0 -

    *g2 = exp(-kappaT) * (exp(2 * kappaT) * theta - exp(2 * kappat) * (theta - 2 *
```

```

    *h1 = (exp(kappaT) * theta + exp(kappat) * ((kappat - kappaT - 1) * theta + ka
    *h2 = (exp(kappat) - exp(kappaT)) * (exp(kappat) * (theta - 2 * v0) - exp(kapp
    *wu = (-exp(u * t) + exp(u * T)) / u;
    *w0u = (exp(T * u) * (-t * u + T * u - 1) + exp(t * u)) / SQR(u);
    *wuu = SQR(exp(t * u) - exp(T * u)) / (2 * SQR(u));

    return 0;
}

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

int greeksBS(double x, double y, double K, double T, double r, double divid,
             double *Pxy, double *Py, double *Pxyy, double *Pxyyy)
{
    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
    *Py = (0.25 / (sqrt(2 * M_PI) * SQR(y))) * (exp(x) * exp(-divid * T) * (-2 *
    *Pxyy = (0.5 / (sqrt(2 * M_PI) * y * sqrt(y))) * (exp(x) * exp(-divid * T) * (
    *Pxyyy = (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;

}

/*****
Pricing formula (2.13) page 7
*****/

int ApBGMHeston(double S, double K, double T, double r, double divid,
                double v0, double kappa, double timestep, PnlVect *theta, PnlVect
                PnlVect *rho, double *ptprice, double *ptdelta)
{
    int i;
    double var, a1, a2, b0, b2, w1, w2, alpha, beta, v0t;
    double f0, f1, f2, g1, g2, h1, h2, wu, w0u, wuu;
    double Pxy, Pyy, Pxxy, Pxyy;
    double Pxyhu, Pyyhu, Pxxyhu, Pxyyhu, Pxyhd, Pyyhd, Pxxyhd, Pxyyhd;
    double BS_put_price, BS_put_delta;
    double h;

    a1 = 0.;
    a2 = 0.;
    b0 = 0.;
    w1 = 0.;
    w2 = 0.;
    alpha = 0.;
    beta = 0.;
    v0t = v0;
    var = 0.;
    h = 0.01;

    /*****
    Explicit computations for Picewise constant parameter case see page 7
    *****/

    for (i = 0; i < (int)(T / timestep); i++)
    {
        expansion_terms(kappa, v0, GET(theta, i), ((double)i * timestep), ((double)
        -kappa, &f0, &f1, &f2, &g1, &g2, &h1, &h2, &wu, &w0u, &wuu

        a1 += wu * w1 + GET(rho, i) * GET(vovol, i) * f1;
    }

```

```

        a2 += wu * alpha + GET(rho, i) * GET(vovol, i) * w0u * w1 + SQR(GET(rho, i) * GET(vovol, i));
        b0 += wu * beta + wuu * w2 + SQR(GET(vovol, i)) * f0;
        alpha += GET(rho, i) * GET(vovol, i) * w1 / 4 + SQR(GET(rho, i) * GET(vovol, i));
        beta += wu * w2 + SQR(GET(vovol, i)) * g2;
        w1 += GET(rho, i) * GET(vovol, i) * h1;
        w2 += SQR(GET(vovol, i)) * h2;
        v0t = exp(-kappa / 4) * (v0t - GET(theta, i)) + GET(theta, i);

        var += v0t / 4;
    }

    b2 = 0.5 * SQR(a1);

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

    /*BS put price*/
    pnl_cf_put_bs(S, K, T, r, divid, sqrt(var / T), &BS_put_price, &BS_put_delta);

    /* Put Price given by formula (2.13) page 7*/
    *ptprice = BS_put_price + a1 * Pxy + a2 * Pxyy + b0 * Pyy + b2 * Pxyyy;
    /* Put Delta */
    *ptdelta = BS_put_delta + 0.5 * (a1 * (Pxyhu - Pxyhd) + a2 * (Pxyyhu - Pxyyhd));

    return OK;
}

static int ApBGMTTimeHeston(double S, NumFunc_1 *p, double T, double r, double kappa, double timestep, PnlVect *theta, PnlVect *rho, double *ptprice, double *ptdelta)
{
    double K;

    K = p->Par[0].Val.V_PDDOUBLE;

    ApBGMHeston(S, K, T, r, divid, v0, kappa, timestep, theta, vovol, rho, ptprice);
    if ((p->Compute) == &Call)
    {

```

```

        *ptdelta = 1 + *ptdelta;
        *ptprice = (S - K * exp(-r * T)) + *ptprice;
    }
    return OK;
}

int CALC(AP_BGM_TimeHeston)(void *Opt, void *Mod, PricingMethod *Met)
{
    int status;
    double r, divid;
    PnlVect *theta, *vovol, *rho;
    PnlMat *all_params;
    TYPEOPT *ptOpt = (TYPEOPT *)Opt;
    TYPEMOD *ptMod = (TYPEMOD *)Mod;

    r = log(1. + ptMod->R.Val.V_DOUBLE / 100.);
    divid = log(1. + ptMod->Divid.Val.V_DOUBLE / 100.);

    all_params = pnl_mat_create_from_file(ptMod->TimeDepParameters.Val.V_FILENAME);
    theta = pnl_vect_create(0);
    vovol = pnl_vect_create(0);
    rho = pnl_vect_create(0);
    pnl_mat_get_col(theta, all_params, 0);
    pnl_mat_get_col(vovol, all_params, 1);
    pnl_mat_get_col(rho, all_params, 2);

    status = ApBGMTimeHeston(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->MeanReversion.Val.V_PDOUBLE,
                             ptMod->TimeStep.Val.V_PDOUBLE,
                             theta, vovol, rho,
                             &(Met->Res[0].Val.V_DOUBLE),
                             &(Met->Res[1].Val.V_DOUBLE)
    );

    /* Do not free all_params until you don't use theta, vovol and rho */
    pnl_mat_free(&all_params);
    pnl_vect_free(&theta);
    pnl_vect_free(&vovol);
}

```

```

    pnl_vect_free(&rho);

    return status;
}

static int CHK_OPT(AP_BGM_TimeHeston)(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->HelpFilenameHint = "AP_BGM_TimeHeston";

    return OK;
}

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

```