

[Help](#)

```

#include "hes1d_std.h"
#include <pnl/pnl_mathtools.h>
#include <pnl/pnl_root.h>

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2012+2) //The "#els
static int CHK_OPT(AP_Asymptotics_ImpliedVolatility)(void *Opt, void *Mod)
{
    return NONACTIVE;
}
int CALC(AP_Asymptotics_ImpliedVolatility)(void *Opt, void *Mod, PricingMethod *
{
    return AVAILABLE_IN_FULL_PREMIA;
}
#else

//Large-Time Implied Volatility
//A Rate function
static double Vfun(double kappa, double theta, double sigma, double rho, double
{
    double res;
    res = kappa * theta * (kappa - rho * sigma * p -
                          sqrt(SQR(kappa - rho * sigma * p) + SQR(sigma) * (p -
                          ) / SQR(sigma);

    return res;
}
/* static double VDerivee(double kappa, double theta, double sigma, double rho,d
* {
* double res;
* res = kappa*theta*( - rho*sigma + ( rho*sigma*(kappa - rho*sigma*p) +SQR(sigm
* )/SQR(sigma);
*
* return res;
* } */

static double VSeconde(double kappa, double theta, double sigma, double rho, do
{
    double res;

```

```

    res = kappa * theta * (SQR(sigma) * (1.0 - rho * rho) / sqrt(SQR(kappa - rho *
        SQR(rho * sigma * (kappa - rho * sigma * p) + SQR(sigma
        ) / SQR(sigma);

    return res;
}

//P* gives ths Legendre Transform of V: It is defined as the unique solution of
static double pstar(double kappa, double theta, double sigma, double rho, double
{
    double res;
    res = sigma - 2.0 * kappa * rho + (kappa * theta * rho + x * sigma) *
        sqrt((SQR(sigma) + SQR(2.0 * kappa) - 4.0 * kappa * rho * sigma) / (SQR(
    res /= 2.0 * sigma * (1 - rho * rho);

    return res;
}

// V* (x) = x p*(x) - V(p*(x))
static double Vstar(double kappa, double theta, double sigma, double rho, double
{
    double p = pstar(kappa, theta, sigma, rho, v0, x) ;

    return (x * p - Vfun(kappa, theta, sigma, rho, v0, p));
}

//The functions ABS and A are needed for the second term of the expansion of the
static double A_BS(double x, double sigma, double a)
{
    double res;
    if (x != SQR(sigma) / 2.0 && x != -SQR(sigma) / 2.0)
        res = exp(a * (SQR(2.0 * x / SQR(sigma)) - 1.0) / 8.0) * CUB(sigma) / (SQR(x
    else
        res = (a / 2.0 - 1.0) / sigma ;

    return res;
}

static double Afunc(double kappa, double theta, double sigma, double rho, double
{

```

```

double px = pstar(kappa, theta, sigma, rho, v0, x);
double V = Vfun(kappa, theta, sigma, rho, v0, px);
double V2 = VSeconde(kappa, theta, sigma, rho, v0, px);
double U, d;
double res;

d = sqrt(SQR(kappa - rho * sigma * px) + SQR(sigma) * (px - SQR(px)));

U = exp(2.0 * kappa * theta * log(2.0 * d / (kappa - rho * sigma * px + d)) /

res = U / (sqrt(V2) * px * (px - 1.0));

return res;
}

int ApIVAsymptoticsHeston(double S0, NumFunc_1 *Payoff, double T, double r, dou
{
double sigmaInfty, a1;
double kappaBar = kappa - rho * sigma;
double barTheta = kappa * theta / kappaBar;
double Vx;
double A, Abs;
double x, K;

K = Payoff->Par[0].Val.V_PDDOUBLE;
x = log(K * exp(-(r - div) * T) / S0) / T;
if (x == 0)
    x = 0.001;

Vx = Vstar(kappa, theta, sigma, rho, v0, x);
pstar(kappa, theta, sigma, rho, v0, x);

sigmaInfty = 2.0 * (2.0 * Vx - x);

if (x > -theta / 2.0 && x < barTheta / 2.0)
    sigmaInfty += 4.0 * sqrt(SQR(Vx) - Vx * x);
else
    sigmaInfty -= 4.0 * sqrt(SQR(Vx) - Vx * x);

```

[illegible]

```

        ptMod->Rho.Val.V_PDOUBLE,
        &(Met->Res[0].Val.V_DOUBLE)
    );
}

}

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

        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(AP_Asymptotics_ImpliedVolatility) =
{
    "AP_Asymptotics_ImpliedVolatility",
    {" ", PREMIA_NULLTYPE, {0}, FORBID}},
    CALC(AP_Asymptotics_ImpliedVolatility),
    { {"Asymptotics for Implied Volatility ", DOUBLE, {100}, FORBID},
      {" ", PREMIA_NULLTYPE, {0}, FORBID}
    },
    CHK_OPT(AP_Asymptotics_ImpliedVolatility),
    CHK_ok,
    MET(Init)
};

```