

[Help](#)

```

#include "stein1d_std.h"
#include "error_msg.h"

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

// Computation of d1
static double D1(double t, double x, double et, double T, double K, double r, do
{
    return ((x - K + (r - divid) * (T - t) + 1. / 2 * (et * et)) / et);
}
// Computation of s2
static double D2(double t, double x, double et, double T, double K, double r, d
{
    double d2;
    d2 = D1(t, x, et, T, K, r, divid) - et;
    return (d2);
}

// Calcul of E[ c(t,v)_[t,T] ] //
static double g1_c_stein(double t, double x, double v, double T, double a, doub
{
    double nouv_c;
    nouv_c = -c * (0.4e1 * (b * b) * a * v * exp((2 * b * t)) * t - 0.4e1 * (b *
    return (nouv_c);
}

// Expected value of <M(t,v)>T in Stein and Stein model //
static double esperance_stein(double t, double x, double v, double T, double a,

```

```

{
    double nouv_EM;
    nouv_EM = (a * a + c * c / b / 0.2e1) * (T - t) + 0.2e1 * a * (v - a) * (0.1e1
    return (nouv_EM);
}

// Variance of <M(t,v)>T in Stein and Stein model //
static double variance_stein(double t, double x, double v, double T, double a, d
{
    double va;
    va = - (c * c) * ((5 * c * c) - (4 * v * v * b) + 0.8e1 * (c * c) * exp((2 * b
    return (va);
}

//a long term run
//b speed of mean reversion
//c vol of vol
int ApAntonelliScarlattiStein(double S, NumFunc_1 *p, double T, double r, doubl
{
    int flag_call;
    double K, x, t, price, delta, divid;

    double EM, VarM, cT, d10, d20, g01, g02, g1;
    double d1, d2, co, dg0, dg1;

    divid = 0;
    r = r - divid1;
    K = p->Par[0].Val.V_PDOUBLE;

    //Log trasformation
    K = log(K);
    x = log(S);
    t = 0.;

    //Trasformation in variance
    //a=sqrt(a);
    //v=sqrt(v);

    if ((p->Compute) == &Call)

```

```

    flag_call = 1;
else
    flag_call = 0;

//Pricing
EM = esperance_stein(t, x, v, T, a, b, c, r, K, rho);
VarM = variance_stein(t, x, v, T, a, b, c, r, K, rho);
cT = g1_c_stein(t, x, v, T, a, b, c, r, K, rho);
d10 = D1(t, x, sqrt((1 - rho * rho) * EM), T, K, r, divid);
d20 = D2(t, x, sqrt((1 - rho * rho) * EM), T, K, r, divid);
g01 = exp(x) * cdf_nor(d10) - exp(K - r * (T - t)) * cdf_nor(d20); // g0 term
g02 = exp(K - r * (T - t)) / (8 * pow(EM, 1.5)) * (d20 * d10 - 1.) * pnl_normal_density(d20);
g1 = -exp(K - r * (T - t)) / EM * d20 * pnl_normal_density(d20) * cT; // g1 term

//Hedging
d1 = D1(t, x, sqrt(EM), T, K, r, divid);
d2 = D2(t, x, sqrt(EM), T, K, r, divid);

co = g1_c_stein(t, x, v, T, a, b, c, r, K, rho);

// h0 term //
dg0 = exp(x) * cdf_nor(d1);

//h1 term//
dg1 = -exp((double)K - r * (T - t)) * (1. - (d2) * (d2)) * pnl_normal_density(d2);

//Call case
if (flag_call == 1)
{
    price = g01 + g02 + rho * g1;
    delta = (dg0 + rho * dg1) * exp(-x);
} //Put case
else
{
    price = g01 + g02 + rho * g1 - S + exp(K - r * T);
    delta = (dg0 + rho * dg1) * exp(-x) - 1;
}

/* Price*/
*ptprice = price * exp(-divid1 * T);

```

```

    /* Delta */
    *ptdelta = delta * exp(-divid1 * T);

    return OK;
}

int CALC(AP_AntonelliScarlatti_Stein)(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-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 ApAntonelliScarlattiStein(ptMod->S0.Val.V_PDOUBLE,
                                           ptOpt->PayOff.Val.V_NUMFUNC_1,
                                           ptOpt->Maturity.Val.V_DATE - ptMod->T.Val
                                           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->Res[0].Val.V_DOUBLE),
                                           &(Met->Res[1].Val.V_DOUBLE)
                                           );
    }
}

static int CHK_OPT(AP_AntonelliScarlatti_Stein)(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;
    }

    return OK;
}

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