

[Help](#)

```
#include "
href../../mod/nig1d/nig1d_std/nig1d_std_h_src.pdfnig1d_std.h"

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

static double lmin = 1.416;
static double alpha, beta, delta, mu, r, divid, T, S0, strike;
static double uu, vv, ww, rho1, rho2;
static long int points;

// ===== SPM integral =====
static double NIG_spmlam(double e)
{
    double arg, lamw;
    int i;

    arg = 2.0 / 3.0 * pow(4.0 * ww / e / e, 1.0 / 3.0);
    if (ww > 0.001)
    {
        lamw = arg; //w=LambertW(arg)
        if (arg > 3)
        {
            lamw = log(arg) - log(log(arg));
        }
        for (i = 0; i < 5; i++)
        {
            lamw = lamw + lamw / (1 + lamw) * (arg - lamw * exp(lamw)) / (lamw * e
        }
        lamw = sqrt(3.0 / 2.0 / ww * lamw);
    }
```

```

    }
else
    {
        lamw = sqrt(sqrt(arg));
    }
return lamw;
}

static double NIG_spmfun(double x)
{
    double z = x * x;
    double zuu = uu * (1 + z);
    double zvv = vv * vv * z * (2 + z);
    double fr1 = (uu + rho1 * (1 + z)) / ((rho1 + zuu) * (rho1 + zuu) + zvv);
    double fr2 = (uu + rho2 * (1 + z)) / ((rho2 + zuu) * (rho2 + zuu) + zvv);

    return exp(-ww * z) * (fr1 - fr2) * 2.0 * vv / sqrt(2 + z);
}

static double NIG_pereval(double eps)
{
    double res, s1, s2, v1, v2, x, h;

    long int i, n;

    double e = eps / 2.0;

    double lambda = NIG_spmlam(e);

    if (lambda < lmin)
    {
        lambda = lmin;
    };

    h = lambda / 2.0;
    s1 = NIG_spmfun(0) + NIG_spmfun(lambda);
    s2 = NIG_spmfun(h);
    v2 = h * (s1 + 4.0 * s2) / 3.0;
    v1 = 0;

    n = 2;

```

```

while (fabs((v1 - v2) / v2) > e)
{
    v1 = v2;
    s1 += 2.0 * s2;
    s2 = 0;
    x = h / 2.0;
    for (i = 1; i <= n; i++)
    {
        s2 += NIG_spmfun(x);
        x += h;
    }
    h /= 2.0;
    n *= 2.0;
    v2 = h * (s1 + 4.0 * s2) / 3.0;
};

n = ceil(n / 2.0) + 1;
points = n;
res = v2;

return res;
}
static int NIG_inteput(int ifCall, double spot, double strk, double ti, double r)
{
    double factr, term, xx, logs, logk, fac;
    double u, v, edtaur, muadd, res;
    double sig2 = sigma * sigma;

    T = ti;
    r = ri;
    divid = dividi;
    S0 = spot;
    strike = strk;

    alpha = sqrt(theta * theta + sig2 / kappa) / sig2;
    beta = theta / sig2;
    delta = sigma / sqrt(kappa);
    logs = log(spot);
    logk = log(strk);
    xx = logs - logk;

```

```

muadd = delta * (sqrt(alpha * alpha - (beta + 1) * (beta + 1)) - sqrt(alpha *
mu = r - divid + muadd;

u = alpha * (xx + T * mu);
v = alpha * T * delta;
ww = sqrt(u * u + v * v);
uu = u / ww;
vv = v / ww;

rho1 = beta / alpha;
rho2 = (beta + 1) / alpha;

edtaur = exp(-T * r);

fac = exp(v * sqrt(1 - rho1 * rho1)) / M_PI;
factr = fac * exp(-rho1 * u - ww);

term = 0;

if (uu < -rho1)
{
    term = edtaur;
};
if (uu < -rho2)
{
    term = edtaur - exp(xx);
};

res = NIG_pereval(eps);
res = strk * (term + edtaur * factr * res);

if (ifCall)
{
    res = res + spot - strike * edtaur;
}
*ptprice = res;

return OK;
}

```

```

int CALC(CF_spmNIG)(void *Opt, void *Mod, PricingMethod *Met)
{
    TYPEOPT *ptOpt = (TYPEOPT *)Opt;
    TYPEMOD *ptMod = (TYPEMOD *)Mod;
    NumFunc_1 *p;
    double r, divid, tt, s0, strk, sigma, theta, kappa;
    int res;
    int ifCall;

    r = log(1. + ptMod->R.Val.V_DOUBLE / 100.);
    divid = log(1. + ptMod->Divid.Val.V_DOUBLE / 100.);
    tt = ptOpt->Maturity.Val.V_DATE - ptMod->T.Val.V_DATE;
    s0 = ptMod->S0.Val.V_PDOUBLE;
    p = ptOpt->PayOff.Val.V_NUMFUNC_1;
    ifCall = ((p->Compute) == &Call);
    strk = p->Par[0].Val.V_DOUBLE;
    sigma = ptMod->Sigma.Val.V_PDOUBLE;
    theta = ptMod->Theta.Val.V_DOUBLE;
    kappa = ptMod->Kappa.Val.V_SPDOUBLE;

    res = NIG_inteput(ifCall, s0, strk, tt, r, divid, sigma, theta, kappa, 1e-5, &

    return res;
}

static int CHK_OPT(CF_spmNIG)(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;
    }
}

```

```

        Met->HelpFilenameHint = "cf_spm_nig";
    }

    return OK;
}

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

```