

[Help](#)

```
#include "
href../../../../mod/bs2d/bs2d_std2d/bs2d_std2d_h_src.pdfbs2d_std2d.h"
#include "pnl/pnl_cdf.h"
#define PRECISION 1.0e-7 /*Precision for the localization of FD methods*/

static int CallMaxAn(double s1, double s2, double k, double t,
                    double r, double divid1, double divid2,
                    double sigma1, double sigma2, double rho,
                    double *ptprice, double *ptdelta1, double *ptdelta2)
{
    double b1, b2, sigma, rho1, rho2, d, d1, d2, norm1, norm2, norm3;

    b1 = r - divid1;
    b2 = r - divid2;
    sigma = sqrt(SQR(sigma1) + SQR(sigma2) - 2.0 * rho * sigma1 * sigma2);
    if (((sigma - PRECISION) <= 0) && ((rho + PRECISION) >= 1))
    {
        if ((s1 * exp(-divid1 * t)) >= (s2 * exp(-divid2 * t)))
        {
            pnl_cf_call_bs(s1, k, t, r, divid1, sigma1, ptprice, ptdelta1);
            *ptdelta2 = 0.;
        }
        else
        {
            pnl_cf_call_bs(s2, k, t, r, divid2, sigma2, ptprice, ptdelta2);
            *ptdelta1 = 0.;
        }
    }
    else
    {
        rho1 = (sigma1 - rho * sigma2) / sigma;
        rho2 = (sigma2 - rho * sigma1) / sigma;
        d = (log(s1 / s2) + (b1 - b2 + SQR(sigma) / 2.0) * t) / (sigma * sqrt(t));
        d1 = (log(s1 / k) + (b1 + SQR(sigma1) / 2.0) * t) / (sigma1 * sqrt(t));
        d2 = (log(s2 / k) + (b2 + SQR(sigma2) / 2.0) * t) / (sigma2 * sqrt(t));

        norm1 = pnl_cdf2nor(d1, d, rho1);
        norm2 = pnl_cdf2nor(d2, -d + sigma * sqrt(t), rho2);
    }
}
```

```

        norm3 = pnl_cdf2nor(-d1 + sigma1 * sqrt(t), -d2 + sigma2 * sqrt(t), rho);

        /*Price*/
        *ptprice = s1 * exp((b1 - r) * t) * norm1 + s2 * exp((b2 - r) * t) * norm2;

        /*Deltas*/
        *ptdelta1 = exp((b1 - r) * t) * norm1;
        *ptdelta2 = exp((b2 - r) * t) * norm2;
    }
    return 0;
}

int CALC(CF_CallMax)(void *Opt, void *Mod, PricingMethod *Met)
{
    TYPEOPT *ptOpt = (TYPEOPT *)Opt;
    TYPEMOD *ptMod = (TYPEMOD *)Mod;
    double r, divid1, divid2;

    r = log(1. + ptMod->R.Val.V_DOUBLE / 100.);
    divid1 = log(1. + ptMod->Divid1.Val.V_DOUBLE / 100.);
    divid2 = log(1. + ptMod->Divid2.Val.V_DOUBLE / 100.);

    return CallMaxAn(ptMod->S01.Val.V_PDOUBLE, ptMod->S02.Val.V_PDOUBLE, (ptOpt->P
        ptOpt->Maturity.Val.V_DATE - ptMod->T.Val.V_DATE,
        r, divid1, divid2,
        ptMod->Sigma1.Val.V_PDOUBLE, ptMod->Sigma2.Val.V_PDOUBLE, ptM
        &(Met->Res[0].Val.V_DOUBLE), &(Met->Res[1].Val.V_DOUBLE), &(M

}

static int CHK_OPT(CF_CallMax)(void *Opt, void *Mod)
{
    return strcmp(((Option *)Opt)->Name, "CallMaximumEuro");
}

static int MET(Init)(PricingMethod *Met, Option *Opt)
{
    if (Met->init == 0)
    {
        Met->init = 1;
    }
}

```

```

    return OK;
}

PricingMethod MET(CF_CallMax) =
{
    "CF_CallMax",
    {{" ", PREMIA_NULLTYPE, {0}, FORBID}},
    CALC(CF_CallMax),
    { {"Price", DOUBLE, {100}, FORBID}, {"Delta1", DOUBLE, {100}, FORBID} , {"Delt
        {" ", PREMIA_NULLTYPE, {0}, FORBID}
    },
    CHK_OPT(CF_CallMax),
    CHK_ok,
    MET(Init)
} ;

```