Help

```
#include  "bs2d_std2dg.h"
#include "pnl/pnl_cdf.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2013+2) //The "#els
static int CHK_OPT(AP_Spread_HurdZhou_bs)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(AP_Spread_HurdZhou_bs)(void *Opt, void *Mod, PricingMethod *Met)
{
  return AVAILABLE_IN_FULL_PREMIA;
}
#else

static double margrabe(double x1, double x2, double q1, double q2, double s1, do
                       double rho, double k, double t, double r, double *delta1,
{

  //Margrabe formula, K=0
  double s = sqrt(s1 * s1 + s2 * s2 - 2 * rho * s1 * s2);
  double st = s * sqrt(t);
  double d1 = (log(x1 / x2) + (q2 - q1 + s * s / 2) * t) / st;
  double d2 = d1 - st;
  if (delta1 != 0)
    {
      *delta1 = exp(-q1 * t) * pnl_cdfnor(d1);
    }
  if (delta2 != 0)
    {
      *delta2 = -exp(-q2 * t) * pnl_cdfnor(d2);
    }
  return x1 * exp(-q1 * t) * pnl_cdfnor(d1) - x2 * exp(-q2 * t) * pnl_cdfnor(d2)
}


static dcomplex characteristic(dcomplex u1, dcomplex u2, double q1, double q2, d
                               double rho, double r, double t)
{
  dcomplex T1, T2, T3, T4, a1, a2, I;
```

```
  a1 = CRmul(u1, s1);
  a2 = CRmul(u2, s2);
  I.r = 0.0;
  I.i = 1.0;
  T1 = Cmul(I, Cadd(RCmul(r - q1, u1), RCmul(r - q2, u2)));
  T2 = RCmul(rho, Cmul(a1, a2));
  T3 = Cmul(a1, Cadd(a1, CRmul(I, s1)));
  T4 = Cmul(a2, Cadd(a2, CRmul(I, s2)));
  return Cexp(RCmul(t, Csub(Csub(T1, T2), CRmul(Cadd(T3, T4), 0.5))));
}


static dcomplex p_hat(dcomplex u1, dcomplex u2)
{
  dcomplex T1, T2, T3, I;
  I.r = 0.0;
  I.i = 1.0;
  T1 = Ctgamma(CRsub(Cmul(I, Cadd(u1, u2)), 1.0));
  T2 = Ctgamma(Cmul(Cminus(I), u2));
  T3 = Ctgamma(CRadd(Cmul(I, u1), 1.0));
  return Cmul(T1, Cdiv(T2, T3));
}

static dcomplex big_h(double u1, double u2, double eps1, double eps2, double q1,
                      double rho, double r, double t, double x1, double x2)
{
  dcomplex T1, T2, z1, z2, res, I;
  I.r = 0.0;
  I.i = 1.0;
  z1.r = u1;
  z1.i = eps1;
  z2.r = u2;
  z2.i = eps2;
  res = Cmul(p_hat(z1, z2), characteristic(z1, z2, q1, q2, s1, s2, rho, r, t));
  T1 = Cmul(I, RCmul(x1, z1));
  T2 = Cmul(I, RCmul(x2, z2));
  res = Cmul(res, Cexp(Cadd(T1, T2)));
  return res;
}

static int Spread_HurdZhou_bsAn(double s01, double s02, double K, double t, doub
```

```
                                double sigma1, double sigma2, double rho, double
{

  double u_hat, eps1, eps2;
  unsigned int N;
  double x1, x2;

  double eta, fact;
  unsigned long k1, k2;
  dcomplex z;
  double u1, u2, d11, d22;
  dcomplex res, dzdx1, dzdx2, I;

  // 2D Intregration parameters
  N = 256;
  u_hat = 40;
  eps1 = -5;
  eps2 = 3;

  //Hurd-Zhou for strike =1

  //log-spots:
  I.r = 0.0;
  I.i = 1.0;

  //We take K >0 using Call-Put parity
  if (K < 0)
    {
      double e1 = exp(-divid1 * t), e2 = exp(-divid2 * t);
      Spread_HurdZhou_bsAn(s02, s01, -K, t, r, divid2, divid1, sigma2, sigma1, r
      d11 += e1;
      d22 -= e2;
      if (ptdelta1 != 0)
        {
          *ptdelta1 = d11;
        }
      if (ptdelta2 != 0)
        {
          *ptdelta2 = d22;
        }
      *ptprice = (*ptprice) - s02 * e2 + s01 * e1 - exp(-r * t) * K;
```

```
  }

//Use Margrabe
else if (K == 0)
  {
    printf("strike = 0 ! use margrabe ! \ n");
    *ptprice = margrabe(s01, s02, divid1, divid2, sigma1, sigma2, rho, K, t, r
  }

//when K!=1, changes to have K=1
else if (K != 1)
  {
    Spread_HurdZhou_bsAn(s01 / K, s02 / K, 1, t, r, divid1, divid2, sigma1, si
                         ptprice, ptdelta1, ptdelta2);
    *ptprice = (*ptprice) * K;
  }
//Hurd-Zhou method

else if (K == 1)
  {
    //log-spots definition:
    x1 = log(s01);
    x2 = log(s02);
    //Integration step:
    eta = u_hat * 2 / N;

    res.r = 0.0;
    res.i = 0.0;
    dzdx1.r = 0.0;
    dzdx1.i = 0.0;
    dzdx2.r = 0.0;
    dzdx2.i = 0.0;

    for (k1 = 0; k1 < N; k1++)
      {
        u1 = -u_hat + k1 * eta;
        for (k2 = 0; k2 < N; k2++)
          {
            u2 = -u_hat + k2 * eta;
            z = big_h(u1, u2, eps1, eps2, divid1, divid2, sigma1, sigma2, rho,
            res = Cadd(res, z);
```

```
              dzdx1 = Cadd(dzdx1, Cmul(CRsub(CRmul(I, u1), eps1), z));
              dzdx2 = Cadd(dzdx2, Cmul(CRsub(CRmul(I, u2), eps2), z));
            }
        }

      fact = u_hat / N / M_PI;
      fact = fact * fact;
      fact *= exp(-r * t);
      res = CRmul(res, fact);

      *ptdelta1 = Creal(dzdx1) * fact / s01;
      *ptdelta2 = Creal(dzdx2) * fact / s02;

      *ptprice = Creal(res);
    }
  return OK;
}

int CALC(AP_Spread_HurdZhou_bs)(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.);
  //CallSpread
  if ((ptOpt->PayOff.Val.V_NUMFUNC_2)->Compute == CallSpread2d)
    return Spread_HurdZhou_bsAn(ptMod->S01.Val.V_PDOUBLE, ptMod->S02.Val.V_PDOUB
  else//PutSpread with -strike
    return Spread_HurdZhou_bsAn(ptMod->S01.Val.V_PDOUBLE, ptMod->S02.Val.V_PDOUB
}

static int CHK_OPT(AP_Spread_HurdZhou_bs)(void *Opt, void *Mod)
{
  if ((strcmp(((Option *)Opt)->Name, "CallSpread2dEuro") == 0) || (strcmp(((Opti
    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_Spread_HurdZhou_bs) =
{
  "AP_Spread_HurdZhou_bs",
  {{" ", PREMIA_NULLTYPE, {0}, FORBID}},
  CALC(AP_Spread_HurdZhou_bs),
  { {"Price", DOUBLE, {100}, FORBID}, {"Delta1", DOUBLE, {100}, FORBID} , {"Delt
    {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CHK_OPT(AP_Spread_HurdZhou_bs),
  CHK_ok,
  MET(Init)
} ;
```