Help

```
#include  "
href../../../mod/bs2d/bs2d_std2dg/bs2d_std2dg_h_src.pdfbs2d_std2dg.h"
#include "pnl/pnl_cdf.h"

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

static double brend(double a, double b, double c, double fa, double fb, double f
{
  double u = (b - a) * (fb - fc);
  double v = (b - c) * (fb - fa);
  return b - ((b - a) * u - (b - c) * v) / (u - v) / 2;
}

static double minimizer(double x, double y, double a, double b, double c, double
{
  return a * pnl_cdfnor(y + b * cos(x + p)) - c * pnl_cdfnor(y + d * cos(x)) - k
}

static double minimizer_diff1(double x, double y, double a, double b, double c,
{
  return -a * b * sin(x + p) * pnl_normal_density(y + b * cos(x + p)) + c * d *
}

static double minimizer_diff2(double x, double y, double a, double b, double c,
{
  return a * pnl_normal_density(y + b * cos(x + p)) - c * pnl_normal_density(y +
}

static void maximize(double xi, double yi, double *rx, double *ry, double a, dou
{
```

```
  //maximization : using gradient with optimal step
  //xi, yi : initial values

  double x, y, gx, gy, l, r, m = 0, p1, p2, p3;
  int iter = 0;
  x = xi;
  y = yi;
  gx = minimizer_diff1(x, y, a, b, c, d, k, p);
  gy = minimizer_diff2(x, y, a, b, c, d, k, p); //gradients
  r = gx * gx + gy * gy;


  while (r > 0.001 && iter < 10000)
    {
      gx = gx / r * 0.01;
      gy = gy / r * 0.01; // normalize gradient

      //Brend method (1D optimization)
      p1 = minimizer(x, y, a, b, c, d, k, p);
      p2 = minimizer(x + 0.5 * gx, y + 0.5 * gy, a, b, c, d, k, p);
      p3 = minimizer(x + gx, y + gy, a, b, c, d, k, p);
      l = brend(0, 0.5, 1, p1, p2, p3);

      l = (l > -1) ? a : -1;
      l = (l > 1) ? 1 : a;
      x = x + l * gx;
      y = y + l * gy;

      if (minimizer(x, y, a, b, c, d, k, p) > m)
        {
          (*rx) = x;
          (*ry) = y;
        }
      else if (minimizer(x, y, a, b, c, d, k, p) > m + 0.001) break;
      gx = minimizer_diff1(x, y, a, b, c, d, k, p);
      gy = minimizer_diff2(x, y, a, b, c, d, k, p);
      r = gx * gx + gy * gy;
      iter++;
    }
  if (iter >= 100000) printf("Gradient optimality condition not satisfied\ n");
}
```

```
static int Spread_CarmonaAn(double s01, double s02, double K, double t, double r
                            double sigma1, double sigma2, double rho, double *pt
{
  double rt = sqrt(t);
  double e1 = exp(-divid1 * t), e2 = exp(-divid2 * t);
  double a = s01 * e1, b = sigma1 * rt, c = s02 * e2, d = sigma2 * rt, kappa = K
  double phi = acos(rho);
  double res;
  double theta = 1.5 * M_PI, dstar = 0;

  maximize(theta, dstar, &theta, &dstar, a, b, c, d, kappa, phi);
  res = minimizer(theta, dstar, a, b, c, d, kappa, phi);

  *ptdelta1 = exp(-divid1 * t) * pnl_cdfnor(dstar + b * cos(theta + phi));
  *ptdelta2 = -exp(-divid2 * t) * pnl_cdfnor(dstar + d * cos(theta));

  if (res < 0) res = 0.;
  *ptprice = res;


  return OK;
}

int CALC(AP_Spread_Carmona)(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_CarmonaAn(ptMod->S01.Val.V_PDOUBLE, ptMod->S02.Val.V_PDOUBLE,
  else//PutSpread with -strike
    return Spread_CarmonaAn(ptMod->S01.Val.V_PDOUBLE, ptMod->S02.Val.V_PDOUBLE,
}

static int CHK_OPT(AP_Spread_Carmona)(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_Carmona) =
{
  "AP_Spread_Carmona",
  {{" ", PREMIA_NULLTYPE, {0}, FORBID}},
  CALC(AP_Spread_Carmona),
  { {"Price", DOUBLE, {100}, FORBID}, {"Delta1", DOUBLE, {100}, FORBID} , {"Delt
    {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CHK_OPT(AP_Spread_Carmona),
  CHK_ok,
  MET(Init)
} ;
```