Help

```
#include  "schwartztrolle_stdg.h"
#include "pnl/pnl_mathtools.h"
#include "pnl/pnl_vector.h"
#include "pnl/pnl_matrix.h"
#include "pnl/pnl_cdf.h"

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

static double cste12, cste22, h, k1, k2, k12, k21, eta1, eta2, inter11, inter13,
static int Ntau;
static dcomplex u, m, n1, n2;

static void gauss_legendre(double x1, double x2, double x[], double w[], int n)
{

  int m, j, i;
  double z1, z, xm, xl, pp, p3, p2, p1;
  m = (n + 1) / 2;
  xm = 0.5 * (x2 + x1);
  xl = 0.5 * (x2 - x1);
  for (i = 1; i <= m; i++)
    {
      z = cos(3.141592654 * (i - 0.25) / (n + 0.5));
      do
        {
          p1 = 1.0;
          p2 = 0.0;
          for (j = 1; j <= n; j++)
            {
              p3 = p2;
```

```
              p2 = p1;
              p1 = ((2.0 * j - 1.0) * z * p2 - (j - 1.0) * p3) / j;
            }
          pp = n * (z * p1 - p2) / (z * z - 1.0);
          z1 = z;
          z = z1 - p1 / pp;
        }
      while (fabs(z - z1) > 3e-11);

      x[i - 1] = xm - xl * z;
      x[n - i] = xm + xl * z;
      w[i - 1] = 2.0 * xl / ((1.0 - z * z) * pp * pp);
      w[n - i] = w[i - 1];
    }
}

static void ODE_RK4()
{

  int i;
  dcomplex mk1, mk2, mk3, mk4, n1k1, n1k2, n1k3, n1k4, n2k1, n2k2, n2k3, n2k4;
  dcomplex cste11, cste13, cste21, cste23;
  dcomplex Cbuff; // Buffer to simplify the complex operations

  m.r = 0.;
  m.i = 0.;
  n1.r = 0.;
  n1.i = 0.;
  n2.r = 0.;
  n2.i = 0.;

  cste11 = RCadd(-k1, CRmul(u, inter11));
  cste13 = Cmul(u, CRmul(CRadd(u, -1.), inter13));
  cste21 = RCadd(-k2, CRmul(u, inter21));
  cste23 = Cmul(u, CRmul(CRadd(u, -1.), inter23));

  for (i = 0; i < Ntau; i++)
    {

      //mk1=h*(n1*eta1+n2*eta2);
      mk1 = RCmul(h, Cadd(CRmul(n1, eta1), CRmul(n2, eta2)));
```

```
    //n1k1 = h*(-n2*k21+n1*cste11+n1*n1*cste12+cste13);
    n1k1 = RCmul(h, Cadd(Cadd(CRmul(n2, -k21) , Cmul(n1, cste11)) , Cadd(CRmul
    //n2k1 = h*(-n1*k12+n2*cste21+n2*n2*cste22+cste23);
    n2k1 = RCmul(h, Cadd(Cadd(CRmul(n1, -k12) , Cmul(n2, cste21)) , Cadd(CRmul


    //mk2=h*((n1+n1k1*.5)*eta1+(n2+n2k1*.5)*eta2);
    mk2 = RCmul(h, Cadd(CRmul(Cadd(n1, CRmul(n1k1, .5)), eta1) , CRmul(Cadd(n2
    //n1k2=h*(-(n2+n2k1*.5)*k21+(n1+n1k1*.5)*cste11+(n1+n1k1*.5)*(n1+n1k1*.5)*
    Cbuff = Cadd(CRmul(Cmul(Cadd(n1, CRmul(n1k1, .5)) , Cadd(n1, CRmul(n1k1, .
    n1k2 = RCmul(h, Cadd(CRmul(Cadd(n2, CRmul(n2k1, .5)), -k21), Cadd(Cmul(Cad
    //n2k2=h*(-(n1+n1k1*.5)*k12+(n2+n2k1*.5)*cste21+(n2+n2k1*.5)*(n2+n2k1*.5)*
    Cbuff = Cadd(CRmul(Cmul(Cadd(n2, CRmul(n2k1, .5)) , Cadd(n2, CRmul(n2k1, .
    n2k2 = RCmul(h, Cadd(CRmul(Cadd(n1, CRmul(n1k1, .5)), -k12), Cadd(Cmul(Cad

    //mk3=h*((n1+n1k2*.5)*eta1+(n2+n2k2*.5)*eta2);
    mk3 = RCmul(h, Cadd(CRmul(Cadd(n1, CRmul(n1k2, .5)), eta1) , CRmul(Cadd(n2
    //n1k3=h*(-(n2+n2k2*.5)*k21+(n1+n1k2*.5)*cste11+(n1+n1k2*.5)*(n1+n1k2*.5)*
    Cbuff = Cadd(CRmul(Cmul(Cadd(n1, CRmul(n1k2, .5)) , Cadd(n1, CRmul(n1k2, .
    n1k3 = RCmul(h, Cadd(CRmul(Cadd(n2, CRmul(n2k2, .5)), -k21), Cadd(Cmul(Cad
    //n2k3=h*(-(n1+n1k2*.5)*k12+(n2+n2k2*.5)*cste21+(n2+n2k2*.5)*(n2+n2k2*.5)*
    Cbuff = Cadd(CRmul(Cmul(Cadd(n2, CRmul(n2k2, .5)) , Cadd(n2, CRmul(n2k2, .
    n2k3 = RCmul(h, Cadd(CRmul(Cadd(n1, CRmul(n1k2, .5)), -k12), Cadd(Cmul(Cad

    //mk4=h*((n1+n1k3)*eta1+(n2+n2k3)*eta2);
    mk4 = RCmul(h, Cadd(CRmul(Cadd(n1, n1k3), eta1) , CRmul(Cadd(n2, n2k3), et
    //n1k4=h*(-(n2+n2k3)*k21+(n1+n1k3)*cste11+(n1+n1k3)*(n1+n1k3)*cste12+cste1
    Cbuff = Cadd(CRmul(Cmul(Cadd(n1, n1k3) , Cadd(n1, n1k3)), cste12), cste13)
    n1k4 = RCmul(h, Cadd(CRmul(Cadd(n2, n2k3), -k21), Cadd(Cmul(Cadd(n1, n1k3)
    //n2k4=h*(-(n1+n1k3)*k12+(n2+n2k3)*cste21+(n2+n2k3)*(n2+n2k3)*cste22+cste2
    Cbuff = Cadd(CRmul(Cmul(Cadd(n2, n2k3) , Cadd(n2, n2k3)), cste22), cste23)
    n2k4 = RCmul(h, Cadd(CRmul(Cadd(n1, n1k3), -k12), Cadd(Cmul(Cadd(n2, n2k3)

    //m+=(mk1*.5+mk2+mk3+mk4*.5)/3.;
    m = Cadd(m, CRdiv(Cadd(CRmul(mk1, .5) , Cadd(mk2 , Cadd(mk3 , CRmul(mk4, .
    //n1+=(n1k1*.5+n1k2+n1k3+n1k4*.5)/3.;
    n1 = Cadd(n1, CRdiv(Cadd(CRmul(n1k1, .5) , Cadd(n1k2 , Cadd(n1k3 , CRmul(n
    //n2+=(n2k1*.5+n2k2+n2k3+n2k4*.5)/3.;
    n2 = Cadd(n2, CRdiv(Cadd(CRmul(n2k1, .5) , Cadd(n2k2 , Cadd(n2k3 , CRmul(n
  }
return;
```

```
}

static int ap_schwartztrolle(NumFunc_1 *p, double f0T1, double P0T0, PnlVect *v0
{

  double b1, b2, intpsi0, intpsi1, *weight, *abscissa, g0, g1;
  dcomplex psi, psi0, psi1;
  dcomplex Cbuff; // Buffer to simplify the complex operations
  int n, Nint;
  double v1, sv1, ss1, alpha1, gamma1, v2, sv2, ss2, alpha2, gamma2;
  double rho13, rho15, rho35, rho24, rho26, rho46;
  int flag;
  double K;

  if ((p->Compute) == &Call)
    flag = -1;
  else  if ((p->Compute) == &Put)
    flag = 1;

  K = p->Par[0].Val.V_PDOUBLE;
  v1 = GET(v0, 0);
  v2 = GET(v0, 1);

  eta1 = GET(eta, 0);
  eta2 = GET(eta, 1);

  k1 = GET(kappa, 0);
  k2 = GET(kappa, 1);
  k21 = GET(kappa, 2);
  k12 = GET(kappa, 3);

  ss1 = GET(sigma, 0);
  ss2 = GET(sigma, 1);
  sv1 = GET(sigma, 2);
  sv2 = GET(sigma, 3);

  rho13 = GET(rho, 0);
  rho15 = GET(rho, 1);
  rho35 = GET(rho, 2);
  rho24 = GET(rho, 3);
  rho26 = GET(rho, 4);
```

```
rho46 = GET(rho, 5);

gamma1 = GET(gammac, 0);
gamma2 = GET(gammac, 1);

alpha1 = GET(alpha, 0);
alpha2 = GET(alpha, 1);

if (f0T1 / K > 1.35)
  {
    Nint = 120;
  }
else
  {
    Nint = 60;
  }

weight = (double *)malloc(Nint * sizeof(double));
abscissa = (double *)malloc(Nint * sizeof(double));

intpsi0 = 0;
intpsi1 = 0;
// Setting the values of the global variables
Ntau = 1000;
h = opt_mat / Ntau;
b1 = alpha1 / gamma1 * (1 - exp(-gamma1 * fut_mat));
b2 = alpha2 / gamma2 * (1 - exp(-gamma2 * fut_mat));
inter11 = sv1 * (rho15 * ss1 + rho35 * b1);
inter13 = (ss1 * ss1 + b1 * b1 + 2.*rho13 * ss1 * b1) / 2.;
inter21 = sv2 * (rho26 * ss2 + rho46 * b2);
inter23 = (ss2 * ss2 + b2 * b2 + 2.*rho24 * ss2 * b2) / 2.;
cste12 = sv1 * sv1 / 2.;
cste22 = sv2 * sv2 / 2.;

// Starting the algorithm
u.r = 0.;
u.i = 0.;
ODE_RK4();
Cbuff = Cadd(m, Cadd(CRmul(n1, v1), CRmul(n2, v2)));
psi0 = Cexp(Cadd(Cbuff, CRmul(u, (log(f0T1)))));
```

```
u.r = 1.;
u.i = 0.;
ODE_RK4();
Cbuff = Cadd(m, Cadd(CRmul(n1, v1), CRmul(n2, v2)));
psi1 = Cexp(Cadd(Cbuff, CRmul(u, (log(f0T1)))));

gauss_legendre(0, 50, abscissa, weight, Nint / 2);

for (n = 0; n < Nint / 2; n++)
  {
    u.r = 0.;
    u.i = flag * abscissa[n];
    ODE_RK4();
    Cbuff = Cadd(m, Cadd(CRmul(n1, v1), CRmul(n2, v2)));
    psi = Cexp(Cadd(Cbuff, CRmul(u, (log(f0T1)))));
    intpsi0 += Cimag(Cmul(psi, Cexp(CRmul(CI, -flag * abscissa[n] * log(K)))))


    u.r = 1.;
    u.i = flag * abscissa[n];
    ODE_RK4();
    Cbuff = Cadd(m, Cadd(CRmul(n1, v1), CRmul(n2, v2)));
    psi = Cexp(Cadd(Cbuff, CRmul(u, (log(f0T1)))));
    intpsi1 += Cimag(Cmul(psi, Cexp(CRmul(CI, -flag * abscissa[n] * log(K)))))
  }

gauss_legendre(50, 400, abscissa, weight, Nint / 2);

for (n = 0; n < Nint / 2; n++)
  {
    u.r = 0.;
    u.i = flag * abscissa[n];
    ODE_RK4();
    Cbuff = Cadd(m, Cadd(CRmul(n1, v1), CRmul(n2, v2)));
    psi = Cexp(Cadd(Cbuff, CRmul(u, (log(f0T1)))));
    intpsi0 += Cimag(Cmul(psi, Cexp(CRmul(CI, -flag * abscissa[n] * log(K)))))

    u.r = 1.;
    u.i = flag * abscissa[n];
    ODE_RK4();
    Cbuff = Cadd(m, Cadd(CRmul(n1, v1), CRmul(n2, v2)));
```

```
      psi = Cexp(Cadd(Cbuff, CRmul(u, (log(f0T1)))));
      intpsi1 += Cimag(Cmul(psi, Cexp(CRmul(CI, -flag * abscissa[n] * log(K)))));
    }

  g0 = psi0.r / 2. - intpsi0 / M_PI;
  g1 = psi1.r / 2. - intpsi1 / M_PI;

  /*----Price Future------*/
  *ptprice = P0T0 * flag * (K * g0 - g1);

  free(weight);
  free(abscissa);

  return OK;
}

int CALC(AP_SCHWARTZTROLLE)(void *Opt, void *Mod, PricingMethod *Met)
{
  TYPEOPT *ptOpt = (TYPEOPT *)Opt;
  TYPEMOD *ptMod = (TYPEMOD *)Mod;

  return ap_schwartztrolle(ptOpt->PayOff.Val.V_NUMFUNC_1, ptMod->f0T.Val.V_PDOUB
                           ptMod->kappa.Val.V_PNLVECT,
                           ptMod->sigma.Val.V_PNLVECT,
                           ptMod->rho.Val.V_PNLVECT,
                           ptMod->alpha.Val.V_PNLVECT,
                           ptMod->gammac.Val.V_PNLVECT,
                           ptOpt->Maturity.Val.V_DATE - ptMod->T.Val.V_DATE,
                           ptOpt->FutureMaturity.Val.V_DATE - ptMod->T.Val.V_DAT
                           &(Met->Res[0].Val.V_DOUBLE)
                          );
}

static int CHK_OPT(AP_SCHWARTZTROLLE)(void *Opt, void *Mod)
{
  if ((strcmp(((Option *)Opt)->Name, "CallFuture") == 0) || (strcmp(((Option *)O
    return OK;
  return  WRONG;
}

#endif //PremiaCurrentVersion
```

```
static int MET(Init)(PricingMethod *Met, Option *Opt)
{
  //int type_generator;
  if (Met->init == 0)
    {
      Met->init = 1;
      Met->HelpFilenameHint = "ap_schwartztrolle";
    }

  return OK;
}

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