Help

```
#include "
href../../../mod/merhes1d/merhes1d_std/svj_h_src.pdfsvj.h"
#include "
href../../../mod/merhes1d/merhes1d_std/merhes1d_std_h_src.pdfmerhes1d_std.h"

static double T, sigma, rho, kappa, V0, r, divid, teta, lambda0, m0, v, S, K;
static int func_type = 0;
static int heston = 0, merton = 0;
static double phi = 0.;
static int probadelta = 0;
static int bk, initlog = 0;
static dcomplex lk_1;

static void init_log(void)
{
  initlog = 0;
  bk = 0;
}

static double charact_func(double k)
{
  double X, tau, roeps, u, b, I, eps, eps2;
  dcomplex Ak, Bk, Ck, Dk, Lambdak, z1, z2, z3, zeta, psi_moins, psi_plus, expo,
  dcomplex dlk;

  tau   = T;
  eps   = sigma;
  roeps = rho * eps;
  X     = log(S / K) + (r - divid) * tau;
  eps2  = eps * eps;

  if (func_type == 1)
    {
      u = 1.;
      b = kappa - roeps;
      I = 1.;
    }
  else if (func_type == 2)
    {
```

```
    u = -1.;
    b = kappa;
    I =  0.;
  }
else
  {
    printf("erreur : dans charact_func il faut initialiser func_type a 1 ou 2.
    exit(-1);
  }

if (heston == 1)
  {

    z1   = Complex(k * k, -u * k);
    z2   = Complex(b, -roeps * k);
    z2   = Cmul(z2, z2);

    zeta = Cadd(z2, RCmul(eps2, z1));
    zeta = Csqrt(zeta);

    psi_moins = Complex(b, -roeps * k);
    psi_plus  = RCmul(-1., psi_moins);
    psi_moins = Cadd(psi_moins, zeta);
    psi_plus  = Cadd(psi_plus, zeta);

    expo = Cexp(RCmul(-tau, zeta));
    z3   = Cadd(psi_moins , Cmul(psi_plus, expo));

    Bk = RCmul(-1., z1);
    Bk = Cmul(Bk , Csub(Complex(1., 0), expo));
    Bk = Cdiv(Bk, z3);

    Ak = Cdiv(z3 , RCmul(2., zeta));
    Ak = Clog(Ak);

    if (initlog > 0)
      {
        dlk = Csub(Ak, lk_1);
        if (dlk.i < -M_PI)
          {
            bk = bk + 1;
```

```
          }
        else if (dlk.i > M_PI)
          {
            bk = bk - 1;
          }
        initlog++;
        lk_1 = Ak;
      }
    else
      {
        initlog++;
        lk_1 = Ak;
      }

    Ak = Cadd(Ak, Complex(0., 2 * M_PI * bk));

    Ak = RCmul(2. , Ak);
    Ak = Cadd(RCmul(tau, psi_plus) , Ak);
    Ak = RCmul(-kappa * teta / eps2 , Ak);

  }
else
  {
    Ak = Complex(0., 0.);
    Bk = Complex(-0.5 * tau * k * k , 0.5 * tau * u * k);
  }

if (merton == 1)
  {
    z1 = Complex(-0.5 * v * v * k * k + I * (m0 + 0.5 * v * v) , (m0 + I * v *
    z1 = Cexp(z1);
    z2 = Complex(I, k);
    z2 = RCmul(exp(m0 + 0.5 * v * v)  - 1, z2);
    z2 = Cadd(Complex(1., 0.) , z2);
    Lambdak = Csub(z1, z2);

    Ck = Complex(0., 0.);
    Dk = RCmul(tau, Lambdak);

  }
else
```

```
    {
      Ck = Complex(0., 0.);
      Dk = Complex(0., 0.);
    }

  ans = Cadd(Ak , RCmul(V0, Bk));
  ans = Cadd(ans , Ck);
  ans = Cadd(ans , RCmul(lambda0, Dk));
  ans = Cadd(ans , Complex(0., k * X));
  ans = Cexp(ans);
  ans = Cdiv(ans, Complex(0., k));

  return ans.r;
}

static double charact_func0(double k)
{
  double X, tau, roeps, u, eps, eps2;
  dcomplex Ak, Bk, Ck, Dk, Lambdak, z1, z2, z3, zeta, psi_moins, psi_plus, expo,
  dcomplex dlk;

  tau   = T;
  eps   = sigma;
  roeps = rho * eps;
  X     = log(S / K) + (r - divid) * tau;

  u = kappa - roeps / 2.;

  eps2 = eps * eps;

  if (heston == 1)
    {
      zeta.r = k * k * eps2 * (1. - rho * rho) + u * u + eps2 / 4.;
      zeta.i = 2.*k * roeps * u;
      zeta   = Csqrt(zeta);

      psi_moins = Complex(u, roeps * k);
      psi_plus  = RCmul(-1., psi_moins);
      psi_moins = Cadd(psi_moins, zeta);
      psi_plus  = Cadd(psi_plus, zeta);
```

```
    expo = Cexp(RCmul(-tau, zeta));
    z3   = Cadd(psi_moins , Cmul(psi_plus, expo));

    Bk = RCmul(-(k * k + 0.25) , Csub(Complex(1., 0), expo));
    Bk = Cdiv(Bk, z3);

    Ak = Cdiv(z3 , RCmul(2., zeta));
    Ak = Clog(Ak);

    if (initlog > 0)
      {
        dlk = Csub(Ak, lk_1);
        if (dlk.i < -M_PI)
          {
            bk = bk + 1;
          }
        else if (dlk.i > M_PI)
          {
            bk = bk - 1;
          }
        initlog++;
        lk_1 = Ak;
      }
    else
      {
        initlog++;
        lk_1 = Ak;
      }

    Ak = Cadd(Ak, Complex(0., 2 * M_PI * bk));

    Ak = RCmul(2. , Ak);
    Ak = Cadd(RCmul(tau, psi_plus) , Ak);
    Ak = RCmul(-kappa * teta / eps2 , Ak);
  }
else
  {
    Ak = Complex(0., 0.);
    Bk = Complex(-0.5 * tau * (k * k + 0.25) , 0.);
  }
```

```
  if (merton == 1)
    {
      z1 = Complex(0.5 * m0 - 0.5 * v * v * (k * k - 0.25) , -k * (m0 + 0.5 * v
      z1 = Cexp(z1);
      z2 = Complex(0.5, -k);
      z2 = RCmul(exp(m0 + 0.5 * v * v) - 1. , z2);
      z2 = Cadd(Complex(1., 0.) , z2);
      Lambdak = Csub(z1, z2);

      Ck = Complex(0., 0.);
      Dk = RCmul(tau, Lambdak);
    }
  else
    {
      Ck = Complex(0., 0.);
      Dk = Complex(0., 0.);
    }

  ans = Cadd(Ak , RCmul(V0, Bk));
  ans = Cadd(ans , Ck);
  ans = Cadd(ans , RCmul(lambda0, Dk));
  ans = Cadd(ans , RCmul(X, Complex(0.5, -k)));
  ans = Cexp(ans);
  ans = Cdiv(ans, Complex(k * k + 0.25, 0.));

  if (probadelta == 1)
    {
      ans = Cmul(ans , Complex(0.5, -k));
      ans = RCmul(1. / S , ans);
    }

  return ans.r;
}

static double  probabilities(int n)
{
  double tp;
  int i, ngauss = 10, N = 100;
  double a, b, h = 10., tol = 1.e-12, tpi, tp3;

  init_gauss(ngauss);
```

```
  if (n == 1)
    {
      func_type = 1;
    }
  else
    {
      func_type = 2;
    }

  tp3 = 0.;
  tpi = 1.;
  i   = 0;
  while ((fabs(tpi) > tol) && (i < N))
    {
      a    = i * h;
      b    = a + h;
      tpi  = integrale_gauss(charact_func, a, b);

      tp3 += tpi;
      i++;
    }
  tp = tp3;
  tp  = 0.5 + tp / M_PI;
  tp = 0.5 * (1. - phi) + phi * tp;

  free_gauss();

  return tp;
}

static double  probabilities2(void)
{
  double tp;
  int i, ngauss = 10, N = 1000;
  double a, b, h = 1., tol = 1.e-12, tpi, tp3;

  init_gauss(ngauss);

  tp3 = 0.;
  tpi = 1.;
```

```
  i   = 0;
  while ((fabs(tpi) > tol) && (i < N))
    {
      a    = i * h;
      b    = a + h;
      tpi  = integrale_gauss(charact_func0, a, b);
      tp3 += tpi;
      i++;
    }

  tp = tp3;
  tp = K * tp / M_PI;

  free_gauss();

  return tp;
}

int calc_price_svj(SVJPARAMS *svj, double *ptprice, double *ptdelta)
{
  double proba, proba1, proba2, price, delta;

  K = svj->K;
  S = svj->St0;
  T = svj->T;
  sigma = svj->sigmav;
  V0 = svj->V0;
  teta = svj->theta;
  r = svj->r;
  divid = svj->divid;
  rho = svj->rho;
  kappa = svj->kappa;
  lambda0 = svj->lambda;
  m0 = svj->m0;
  v  = svj->v;
  phi = svj->phi;
  heston = svj->heston;
  merton = svj->merton;

  if (svj->type_f == 1)
    {
```

```
      init_log();
      proba1 = probabilities(1);


      init_log();
      proba2 = probabilities(2);

      price = phi * (S * proba1 * exp(-divid * T) - K * exp(-r * T) * proba2);
      delta = phi * (proba1 * exp(-divid * T));
    }
  else if (svj->type_f == 2)
    {
      init_log();
      probadelta = 0;
      proba = probabilities2();

      price = 0.5 * (1. + phi) * S * exp(-divid * T) + 0.5 * (1. - phi) * K * ex
      probadelta = 1;
      proba = probabilities2();
      delta = 0.5 * (1. + phi) * exp(-divid * T) - proba * exp(-r * T);
    }
  else
    {
      printf("Erreur dans svj.c : parametre svj->type_f inconnu.\ n");
      exit(-1);
    }


  /* Price*/
  *ptprice = price;

  /* Delta */
  *ptdelta = delta;
  return OK;
}
```