Help

```
#include "alsabr11d_std.h"
#include "pnl/pnl_specfun.h"
#include "math/golden.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2010+2) //The "#els
static int CHK_OPT(CF_RogersVeraart1)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(CF_RogersVeraart1)(void *Opt, void *Mod, PricingMethod *Met)
{
  return AVAILABLE_IN_FULL_PREMIA;
}
#else

typedef struct
{
  double z0;
  double sigma0;
  double newgamma;
  double eta;
  double a1;
  double a2;
  double T;
  double r;
  double K;
} alsabr11d_params;

static double integrand_put(double z, void *p)
{
  double log_ptz;
  double c, u, v, q, x, result;
  double log_A, log_B, d1, d2;

  double z0, sigma0, newgamma, eta, a1, a2, T, r, K;

  alsabr11d_params *par = (alsabr11d_params *)p;

  if (z < DBL_EPSILON)
```

```
    {
      return integrand_put(1.1 * DBL_EPSILON, p);
    }
  if (z >= DBL_MAX)
    {
      return 0.0;
    }

  z0 = par->z0;
  sigma0 = par->sigma0;
  newgamma = par->newgamma;
  eta = par->eta;
  a1 = par->a1;
  a2 = par->a2;
  T = par->T;
  r = par->r;
  K = par->K;

  log_ptz = 0.;

  d2 = newgamma / (2 * eta * sqrt(T)) * (log((pow(sigma0, 2. / newgamma) * pow(z
  d1 = d2 + 2 * eta * sqrt(T) / newgamma;

  log_B = 2. / newgamma * log(sigma0) + 1. / newgamma * log(z) + SQR(eta) * T /
  log_A = -r * T + log(K) + pnl_sf_log_erfc(d2 / M_SQRT2) - M_LN2;

  //Log CIR-Density
  c = 2 * a2 / (4.*(1. - exp(-a2 * T)));
  u = c * z0 * exp(-a2 * T);
  v = c * z;
  q = a1 / 2. - 1.;
  x = 2 * sqrt(u * v);
  if (a1 >= 0)
    log_ptz = log(c) - u - v + q / 2.*log(v / u) + log(pnl_bessel_i_scaled(q, x)
  else if (a1 < 0)
    log_ptz = log(c) - u - v + q / 2.*log(v / u) + log(pnl_bessel_i_scaled(fabs(

  result = exp(log_ptz) * (exp(log_A) - exp(log_B));

  return result;
}
```

```
static double integrand_put_0_to_1(double z, void *p)
{
  if (z == 0)
    {
      return 0.;
    }

  else return integrand_put((1 - z) / z, p) / (SQR(z));
}

static double integrand_density(double z, void *p)
{
  double log_ptz = 0.;
  double c, u, v, q, x, result;

  double z0, a1, a2, T;
  alsabr11d_params *par = (alsabr11d_params *)p;

  z0 = par->z0;
  a1 = par->a1;
  a2 = par->a2;
  T = par->T;

  if (z == 0) return 0;

  //Log CIR-Density
  c = 2 * a2 / (4.*(1. - exp(-a2 * T)));
  u = c * z0 * exp(-a2 * T);
  v = c * z;
  q = a1 / 2. - 1.;
  x = 2 * sqrt(u * v);
  if (a1 >= 0)
    log_ptz = log(c) - u - v + q / 2.*log(v / u) + log(pnl_bessel_i_scaled(q, x)
  else if (a1 < 0)
    log_ptz = log(c) - u - v + q / 2.*log(v / u) + log(pnl_bessel_i_scaled(fabs(

  result = exp(log_ptz);

  return result;
}
```

```
static double integrand_density_0_to_1(double z, void *p)
{
  if (z == 0)
    {
      return 0.;
    }

  else return integrand_density((1 - z) / z, p) / (SQR(z));
}

////////////////////////////////////////////////////////////////
int RogersVeraart1(double S0, double z0l, double gam, double etal, NumFunc_1  *p
{
  int neval;
  double price = 0.;
  double abserr, integral, Delta;
  double a1, a2, r, T, K;

  PnlFunc func_pnl;
  alsabr11d_params par;

  pnl_deactivate_mtherr();

  a1 = 2 * (gam - 1) / gam;
  a2 = (2 - gam) * SQR(etal) / gam;
  r = interest_rate;
  T = option_maturity;
  K = p->Par[0].Val.V_PDOUBLE;

  par.z0 = z0l;
  par.sigma0 = sqrt(pow(S0, gam) / z0l);
  par.newgamma = gam;
  par.eta = etal;
  par.a1 = a1;
  par.a2 = a2;
  par.T = T;
  par.r = r;
  par.K = K;

  Delta = 0.;
```

```
  if (a1 < 0)
    {
      func_pnl.F = integrand_density_0_to_1;
      func_pnl.params = &par;

      neval = 100;
      pnl_integration_GK(&func_pnl, 0., 1., 1e-5, 1e-6, &integral, &abserr, &nev

      Delta = exp(-r * T) * K * (1. - integral);
    }

  func_pnl.F = &integrand_put_0_to_1;
  func_pnl.params = &par;

  neval = 100;
  pnl_integration_GK(&func_pnl, 0., 1., 1e-20, 1e-20, &integral, &abserr, &neval

  price = integral + Delta;

  //Call case by parity
  if ((p->Compute) == &Call)
    price = price + S0 - K * exp(-r * T);

  /* Price*/
  *ptprice = price;

  return OK;
}

int CALC(CF_RogersVeraart1)(void *Opt, void *Mod, PricingMethod *Met)
{
  TYPEOPT *ptOpt = (TYPEOPT *)Opt;
  TYPEMOD *ptMod = (TYPEMOD *)Mod;
  int out;
  double r, divid, pseudo_r, option_maturity, option_price;

  option_price = 0.;

  r = log(1. + ptMod->R.Val.V_DOUBLE / 100.);
  divid = log(1. + ptMod->Divid.Val.V_DOUBLE / 100.);
  pseudo_r = r - divid;
```

```
  option_maturity = ptOpt->Maturity.Val.V_DATE - ptMod->T.Val.V_DATE;

  out = RogersVeraart1(ptMod->S0.Val.V_PDOUBLE,
                       ptMod->z0.Val.V_SPDOUBLE,
                       ptMod->gam.Val.V_RGDOUBLE02,
                       ptMod->eta.Val.V_SPDOUBLE,
                       ptOpt->PayOff.Val.V_NUMFUNC_1,
                       option_maturity,
                       pseudo_r,
                       &option_price);

  Met->Res[0].Val.V_DOUBLE = exp(-divid * option_maturity) * option_price;

  return out;

}

static int CHK_OPT(CF_RogersVeraart1)(void *Opt, void *Mod)
{
  if ((strcmp(((Option *)Opt)->Name, "CallEuro") == 0) || (strcmp(((Option *)Opt

    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(CF_RogersVeraart1) =
{
  "CF_RogersVeraart1",
  {   {" ", PREMIA_NULLTYPE, {0}, FORBID}},
  CALC(CF_RogersVeraart1),
```

```
  { {"Price", DOUBLE, {100}, FORBID},
    {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CHK_OPT(CF_RogersVeraart1),
  CHK_ok,
  MET(Init)
};
```