Help

```
#include  "
href../../../mod/alsabr21d/alsabr21d_std/alsabr21d_std_h_src.pdfalsabr21d_std.h"
#include "pnl/pnl_specfun.h"
#include "
href../../../common/math/golden_h_src.pdfmath/golden.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2010+2) //The "#els
static int CHK_OPT(CF_RogersVeraart2)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(CF_RogersVeraart2)(void *Opt, void *Mod, PricingMethod *Met)
{
  return AVAILABLE_IN_FULL_PREMIA;
}
#else

typedef struct
{
  double z0;
  double a1;
  double a2;
  double c1;
  double c2;
  double T;
  double sigma0;
  double r;
  double K;
  double eta;
  double mu;

} alsabr21d_params;


//Generalized HyperGeometric function
static double g(double z, void *p)
{
  double a, b, x, result;
```

```
  double a1, a2, c1, c2, mu;

  alsabr21d_params *par = (alsabr21d_params *)p;

  if (z < 1e-100) return g(2 * 1e-100, par);

  a1 = par->a1;
  a2 = par->a2;
  c1 = par->c1;
  c2 = par->c2;
  mu = par->mu;

  a = -mu / a2;
  b = a1 / 2.;
  x = a2 * z / 2.;

  result = c1 * pnl_sf_hyperg_1F1(a, b, x) + c2 * pnl_sf_hyperg_U(a, b, x);
  return result;
}

//Integration of the Black-Scholes times CIR density for the
//Put case
static double integrand_put(double z, void *p)
{
  double log_ptz = 0.;
  double c, u, v, q, x, result;
  double d1, d2, g_z, log_g_z, log_A, log_B;

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

  alsabr21d_params *par = (alsabr21d_params *)p;

  if (z < 1e-100) return integrand_put(2 * 1e-100, par);
  if (z >= DBL_MAX) return -0.0;

  z0 = par->z0;
  a1 = par->a1;
  a2 = par->a2;
  T = par->T;
  sigma0 = par->sigma0;
  r = par->r;
```

```
  K = par->K;
  eta = par->eta;
  mu = par->mu;

  g_z = g(z, par);

  if (pnl_isinf(g_z) || pnl_isnan(g_z))
    {
      return -0.;
    }

  log_g_z = log(g_z);
  d1 = 1. / (eta * sqrt(T)) * (log(sigma0 * exp(mu * T) / K) + log_g_z + (r + SQ
  d2 = d1 - eta * sqrt(T);

  //Computing log of various arguments
  log_B = log(sigma0) + mu * T + log_g_z + pnl_sf_log_erfc(d1 / M_SQRT2) - M_LN2
  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_A) - exp(log_B)) * exp(log_ptz);

  return result;
}

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

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



////////////////////////////////////////////////////////////////
int RogersVeraart2(double s0, double z0l, double mul, double etal, double a1l, d
{
  int neval;
  double price, abserr;
  PnlFunc func_pnl;
  alsabr21d_params par;

  pnl_deactivate_mtherr();

  par.z0 = z0l;
  par.a1 = a1l;
  par.a2 = a2l;
  par.c1 = c1l;
  par.c2 = c2l;
  par.T = T;
  par.r = r;
  par.K = p->Par[0].Val.V_PDOUBLE;
  par.eta = etal;
  par.mu = mul;
  par.sigma0 = 0.;
  par.sigma0 = s0 / g(z0l, &par);

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

  neval = 200;
  pnl_integration_GK(&func_pnl, 0., 1., 1e-20, 1e-20, &price, &abserr, &neval);

  if ((p->Compute) == &Call)
    price = price + s0 - (par.K) * exp(-r * T);

  /* Price*/
  *ptprice = price;
```

```
  return OK;
}

int CALC(CF_RogersVeraart2)(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 = RogersVeraart2(ptMod->S0.Val.V_PDOUBLE,
                       ptMod->z0.Val.V_SPDOUBLE,
                       ptMod->mu.Val.V_SNDOUBLE,
                       ptMod->eta.Val.V_SPDOUBLE,
                       ptMod->a1.Val.V_SDOUBLE2,
                       ptMod->a2.Val.V_SPDOUBLE,
                       ptMod->c1.Val.V_PDOUBLE,
                       ptMod->c2.Val.V_PDOUBLE,
                       option_maturity,
                       pseudo_r,
                       ptOpt->PayOff.Val.V_NUMFUNC_1,
                       &option_price);

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

  return out;
}

static int CHK_OPT(CF_RogersVeraart2)(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_RogersVeraart2) =
{
  "CF_RogersVeraart2",
  {{" ", PREMIA_NULLTYPE, {0}, FORBID}},
  CALC(CF_RogersVeraart2),
  { {"Price", DOUBLE, {100}, FORBID},
    {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CHK_OPT(CF_RogersVeraart2),
  CHK_ok,
  MET(Init)
};
```