Help

```
extern "C" {
#include  "
href../../../mod/cir1d/cir1d_pad/cir1d_pad_h_src.pdfcir1d_pad.h"
#include "
href../../../common/math/linsys_h_src.pdfmath/linsys.h"
#include "pnl/pnl_cdf.h"
#include "pnl/pnl_integration.h"
#include "pnl/pnl_specfun.h"
#include "pnl/pnl_vector.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2017+2) //The "#els

  static int CHK_OPT(AP_FixedAsian_CIR)(void *Opt, void *Mod)
  {
return NONACTIVE;
  }
  int CALC(AP_FixedAsian_CIR)(void *Opt, void *Mod, PricingMethod *Met)
  {
return AVAILABLE_IN_FULL_PREMIA;
  }
#else
  ////////////////////////////////////////////////////////////////
}

typedef struct { double mean, var, pdfcon, logpdfcon; } ret;

static double BI11(double v, double z) {
  if (z <= 200 && v <= 30) {
    return pnl_bessel_i(v + 1, z) / pnl_bessel_i(v - 1, z);
  } else {
    return 1.0;
  }
}

static double BI31(double v, double z) {
  if (z <= 200 && v <= 30) {
    return pnl_bessel_i(v - 3, z) / pnl_bessel_i(v - 1, z);
  } else {
    return 1.0;
```

```
  }
}

static double BI21(double v, double z) {
  if (z <= 200 && v <= 30) {
    return pnl_bessel_i(v - 2, z) / pnl_bessel_i(v - 1, z);
  } else {
    return 1.0;
  }
}

static double BI01(double v, double z) {
  if (z <= 200 && v <= 30) {
    return pnl_bessel_i(v, z) / pnl_bessel_i(v - 1, z);
  } else {
    return 1.0;
  }
}

double BI(double v, double z) {
  if (z <= 700 && v <= 30) {
    return pnl_bessel_i(v, z);
  } else {
    return exp(z) / sqrt(2 * M_PI * z);
  }
}

static double logBI(double v, double z) {
  if (z <= 500 && v <= 30) {
    return log(BI(v, z));
  } else {
    return z - 0.5 * log(2 * M_PI * z);
  }
}

static ret meanvar(double a, double b, double s0, double st, double sigma, doubl
  double logh, logpdfcon, pdfcon, g, h, i, j, g1, h1, H1, i1, I1, j1, J1, g2,
h2, H2, i2, I2, j2, J2, moment1, moment2, mean, var;
  double dens, bc, delta, ggm, gm, alf, sg2 = pow(sigma, 2);

  bc = 4 * b * sqrt(s0 * st * exp(-b * T)) / (sg2 * (1 - exp(-b * T)));
```

```
  // first moment
  delta = (s0 + st) * (exp(2 * b * T) - 2 * b * T * exp(b * T) - 1);
  alf = 2 * a / sg2;
  gm = BI01(2 * a / sg2, bc) + BI21(2 * a / sg2, bc);
  ggm = BI31(2 * a / sg2, bc) + BI11(2 * a / sg2, bc);
  moment1 = -sg2 / b / b + (sg2 * T * (exp(2 * b * T) - 1) / 2 + delta + (b * T
  mean = moment1;
  // second derivative
  g = 2 * b / (sg2 * (1 - exp(-b * T)));
  h = exp(-2 * b * st / sg2 - g * (s0 + st) * exp(-b * T));
  logh = -2 * b * st / sg2 - g * (s0 + st) * exp(-b * T);
  i = pow(st * exp(b * T) / s0, a / sg2 - 0.5);
  j = exp(logBI(2 * a / sg2 - 1, bc));
  pdfcon = 1 / (g * h * i * j);
  logpdfcon = -log(g) - logh - log(i) - logBI(2 * a / sg2 - 1, bc);
  g1 = 2 * T * exp(-b * T) / pow(1 - exp(-b * T), 2) - 2 / (b * (1 - exp(-b * T)
  h1 = h * ((a * T + s0 + st) / b - 2 * (s0 + st) * T * exp(-b * T) / (1 - exp(-
  H1 = (a * T + s0 + st) / b - 2 * (s0 + st) * T * exp(-b * T) / (1 - exp(-b * T
  i1 = T * (sg2 / 2 - a) * pow(st * exp(b * T) / s0, a / sg2 - 0.5) / b;
  I1 = T * (sg2 / 2 - a) / b;
  j1 = (sqrt(s0 * st) * (T * b + exp(T * b) * (T * b - 2) + 2) * (exp(logBI(2 *
  J1 = (sqrt(s0 * st) * (T * b + exp(T * b) * (T * b - 2) + 2) * exp(logBI(2 * a
  g2 = -(2 * sg2 * exp(T * b) * (1 + exp(2 * T * b) - pow(b * T, 2) - T * b + ex
(pow(b * (exp(T * b) - 1), 3));
  h2 = sg2 * ((s0 + st) * (exp(2 * T * b) * (-2 * b * b * T * T + 2 * T * b - 1)
  H2 = sg2 * ((s0 + st) * (exp(2 * T * b) * (-2 * b * b * T * T + 2 * T * b - 1)
  i2 = T * pow(sigma, 4) * (a / sg2 - 0.5) * ((a * T - 0.5 * T * sg2) / (b * b *
  I2 = T * pow(sigma, 4) * (a / sg2 - 0.5) * ((a * T - 0.5 * T * sg2) / (b * b *
  j2 = ((BI(2 * a / pow(sigma, 2) - 3, bc) + 2 * BI(2 * a / pow(sigma, 2) - 1, b
  J2 = ((exp(logBI(2 * a / pow(sigma, 2) - 3, bc) - logBI(2 * a / pow(sigma, 2)
  moment2 = 2 * (I1 + H1) * (J1 + g1 / g) + 2 * (g1 / g * J1 + H1 * I1) +
(J2 + H2 + I2 + g2 / g);
  var = -2.0 * sg2 * sg2 / pow(b, 4) - sg2 * (1 - exp(b * T) * (1 + 2.0 * b * T)
  if (var < moment2 - pow(moment1, 2)) {
    var = moment2 - pow(moment1, 2);
  }
  sg2 = sigma * sigma;
  bc = 4 * b * sqrt(s0 * st * exp(-b * T)) / (sg2 * (1 - exp(-b * T)));
  dens = 2 * b * exp(logBI(2 * a / pow(sigma, 2) - 1, bc) - 2 * b * (st + s0 * e
  ret t;
  t.mean = mean;
```

```
  t.var = var, t.pdfcon = dens, t.logpdfcon = log(dens);
  return t;
}


static int CIR_FixedAsian(double pseudo_stock, double pseudo_strike, NumFunc_2
{
  double call_or_put;
  double CTtK=0, PTtK=0;


  if ((p->Compute) == &Call_OverSpot2)
    call_or_put = 1;
  else
    call_or_put = 0;

  double pdfcon, bc, bara;
  double precision;
  double K;

  precision=100;
  K=pseudo_strike*t;

  double sg2, x;
  double Price = 0;
  double dens, mean, var, logpdfcon = 0, k, theta, sdlog, meanlog;
  double end, start = 0.00001;
  sg2 = sigma * sigma;
  x = start;
  dens = 0.0;
  while (dens < 0.00000001) {
    bc = 4 * a * sqrt(pseudo_stock * x * exp(-a * t)) / (sg2 * (1 - exp(-a * t))
    dens = 2 * a * exp(logBI(2 * a * b / pow(sigma, 2) - 1, bc) - 2 * a * (x + p
    x += 0.01;
  }
  while (dens >= 0.00000001) {
    bc = 4 * a * sqrt(pseudo_stock * x * exp(-a * t)) / (sg2 * (1 - exp(-a * t))
    dens = 2 * a * exp(logBI(2 * a * b / pow(sigma, 2) - 1, bc) - 2 * a * (x + p
    x += 0.01;
  }
  end = x;
```

```
  double deltaz, deltax = (end) / precision;
  for (x = start; x <= end; x += deltax) {
    ret calculate = meanvar(a * b, a, pseudo_stock, x, sigma, t);
    mean = calculate.mean;
    var = calculate.var;
    pdfcon = calculate.pdfcon;
    logpdfcon = calculate.logpdfcon;
    k = mean * mean / var;
    theta = var / mean;
    sdlog = sqrt(log(var / pow(mean, 2) + 1));
    meanlog = log(mean) - pow(sdlog, 2) / 2;
    deltaz = 5 * sqrt(var) / precision;
    bc = 4 * a * sqrt(pseudo_stock * x * exp(-a * t)) / (sg2 * (1 - exp(-a * t))
    dens = 2 * a * exp(logBI(2 * a * b / pow(sigma, 2) - 1, bc) - 2 * a * (x + p
    if (k > 0) {
  Price += deltax * dens * (k * theta * pnl_sf_gamma_inc_Q(k + 1, K * (1 + 1 / t
    }
  }
  CTtK=Price / t;

  /*  Put Price from Parity*/
  bara=sqrt(a*a+2*sg2);

  PTtK = CTtK + exp(-pseudo_stock * (-2) *(exp(-bara*t)-1)/(a+bara+exp(-bara*t)*

  /*Price*/
  if(call_or_put)
    *ptprice = CTtK;
  else
    *ptprice = PTtK;

  return OK;
}


extern "C" {

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

```
    int return_value;
    double time_spent, pseudo_spot, pseudo_strike;
    double t_0, T_0;

    T_0 = ptMod->T.Val.V_DATE;
    t_0 = (ptOpt->PathDep.Val.V_NUMFUNC_2)->Par[0].Val.V_PDOUBLE;


    if (T_0 < t_0)
      {
        Fprintf(TOSCREEN, "T_0 < t_0, untreated case\ n\ n\ n");
        return_value = WRONG;
      }
    /* Case t_0 <= T_0 */
    else
      {
        time_spent = (ptMod->T.Val.V_DATE - (ptOpt->PathDep.Val.V_NUMFUNC_2)->Pa
        pseudo_spot = (1. - time_spent) * ptMod->r0.Val.V_PDOUBLE;
        pseudo_strike = (ptOpt->PayOff.Val.V_NUMFUNC_2)->Par[0].Val.V_PDOUBLE -

return_value = CIR_FixedAsian(pseudo_spot, pseudo_strike, ptOpt->PayOff.Val.V_NU
      }

    return return_value;
  }

  static int CHK_OPT(AP_FixedAsian_CIR)(void *Opt, void *Mod)
  {
    if ((strcmp(((Option *)Opt)->Name, "AsianCallFixedEuroI") == 0) || (strcmp((
      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_FixedAsian_CIR) =
{
  "AP_FixedAsian_CIR",
  { {" ", PREMIA_NULLTYPE, {0}, FORBID}},
  CALC(AP_FixedAsian_CIR),
  {{"Price", DOUBLE, {100}, FORBID}, {" ", PREMIA_NULLTYPE, {0}, FORBID}},
  CHK_OPT(AP_FixedAsian_CIR),
  CHK_ok,
  MET(Init)
};
}
```