Help

```
#include "hes1d_std.h"
#include "pnl/pnl_cdf.h"
#include "pnl/pnl_finance.h"
#include "pnl/pnl_mathtools.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2014+2) //The "#els
static int CHK_OPT(AP_Larsson)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(AP_Larsson)(void *Opt, void *Mod, PricingMethod *Met)
{
  return AVAILABLE_IN_FULL_PREMIA;
}
#else

//****************************************************************
//                  Page 878
//****************************************************************/

static int HestonZero(double K, double T, double S0, double r, double divid, dou
{

  double SigmaT, Nd1, d1, Nd2, d2, q, mean, sd, bound, DOT;
  int which, status;

  which = 1;
  mean = 0.0;
  sd = 1.0;
  bound = 0.0;
  DOT = exp(-(r - divid) * T);

  /* Right Side Line -10 */
  SigmaT = (v0 - b) * (1 - exp(-a * T)) * (1 / a) + b * T;
  /* Right Side Line -13 */
  d1 = (log(S0 / K) + (r - divid) * T + 0.5 * SigmaT) / sqrt(SigmaT);
  d2 = d1 - sqrt(SigmaT);

  pnl_cdf_nor(&which, &Nd1, &q, &d1, &mean, &sd, &status, &bound);
```

```
  pnl_cdf_nor(&which, &Nd2, &q, &d2, &mean, &sd, &status, &bound);

  /* Right Side Line +15 */
  *H00 = exp(-divid * T) * (S0 * Nd1 - K * D0T * Nd2);

  return 0;
}



/*****************************************************************
                      Page 879
*****************************************************************/

static int expansion_terms(double K, double T, double S0,  double r, double divi
{

  double SigmaT, Q1T, Q2T, Q3T, Q4T, d2, a2, a3, b2, v02, D0T, T2;

  a2 = SQR(a);
  a3 = CUB(a);
  b2 = SQR(b);
  v02 = SQR(v0);
  T2 = SQR(T);
  D0T = exp(-(r) * T);
  /* Page 878 Right Side Line -10 */
  SigmaT = (v0 - b) * (1 - exp(-a * T)) * (1 / a) + b * T;
  /* Page 878 Right Side Line -13 */
  d2 = (log(S0 / K) + (r - divid) * T + 0.5 * SigmaT) / sqrt(SigmaT) - sqrt(Sigm

  /*Page 879 Right Side Line +2 */
  Q1T = (1 / a2) * exp(-a * T) * (2 * b - v0 + a * b * T - a * v0 * T) + (1 / a2

  /* Right Side Line +3 */
  Q2T = 0.5 * (1 / SQR(a2)) * exp(-2 * a * T) * (4 * b * v0 - v02 - 4 * b2 + 6 *
        - 4 * a * b2 * T - a2 * v02 * T2 - 2 * a * v02 * T - a2 * b2 * T2)
        + 0.5 * (1 / SQR(a2)) * exp(-a * T) * (2 * v02 + 2 * a2 * b * v0 * T2 -
            - 2 * a2 * b2 * T2 + 2 * a * v02 * T)
        + 0.5 * (1 / SQR(a2)) * (4 * b * v0 + 4 * a * b2 * T - 2 * a * b * v0 *
```

```
  /* Right Side Line +8 */
  Q3T = (2 / a3) * exp(-a * T) * (2 * a * v0 * T + 2 * v0 - 6 * b - 4 * a * b *
        + (2 / a3) * (6 * b - 2 * v0 - 2 * a * b * T);

  /* Right Side Line +10 */
  Q4T = 0.5 * (1 / a3) * exp(-2 * a * T) * (b - 2 * v0) + 0.5 * (1 / a3) * exp(-
        + 0.5 * (1 / a3) * (2 * v0 - 5 * b + 2 * a * b * T);


  /* Left Side Line -6 */
  *d1H = -0.5 * DOT * K * sigma * rho * d2 * pnl_normal_density(d2) * Q1T * (1 /

  /* Left Side Line -5 */
  *d2H = 0.125 * DOT * K * SQR(sigma) * SQR(rho) * (3 - 6 * SQR(d2) + SQR(SQR(d2
         + 0.25 * DOT * K * SQR(sigma) * SQR(rho) * (3 * d2 - CUB(d2)) * pnl_nor
         + 0.125 * DOT * K * SQR(sigma) * SQR(rho) * (1 - SQR(d2)) * pnl_normal_
         - 0.125 * DOT * K * SQR(sigma) * (1 - SQR(d2)) * pnl_normal_density(d2)
         + 0.125 * DOT * K * SQR(sigma) * d2 * pnl_normal_density(d2) * Q4T * (1

  return 0;
}


static int ApLarssonHeston(double S0, NumFunc_1  *p, double T, double r, double
                           double v0, double a, double b, double sigma, double r
                           double *ptprice)
{
  double K;
  double H00, d1H, d2H;

  K = p->Par[0].Val.V_PDOUBLE;

  HestonZero(K, T, S0, r, divid, a, b, v0, &H00);
  expansion_terms(K, T, S0, r, divid, a, b, v0, sigma, rho, &d1H, &d2H);

  //Call price
  *ptprice = H00 + d1H + d2H;

  return OK;
}
```

```
int CALC(AP_Larsson)(void *Opt, void *Mod, PricingMethod *Met)
{
  TYPEOPT *ptOpt = (TYPEOPT *)Opt;
  TYPEMOD *ptMod = (TYPEMOD *)Mod;
  double r, divid;

  if (ptMod->Sigma.Val.V_PDOUBLE == 0.0)
    {
      Fprintf(TOSCREEN, "BLACK-SHOLES MODEL\ n\ n\ n");
      return WRONG;
    }
  else
    {
      r = log(1. + ptMod->R.Val.V_DOUBLE / 100.);
      divid = log(1. + ptMod->Divid.Val.V_DOUBLE / 100.);

      return ApLarssonHeston(ptMod->S0.Val.V_PDOUBLE,
                             ptOpt->PayOff.Val.V_NUMFUNC_1,
                             ptOpt->Maturity.Val.V_DATE - ptMod->T.Val.V_DATE,
                             r,
                             divid, ptMod->Sigma0.Val.V_PDOUBLE,
                             ptMod->MeanReversion.Val.V_PDOUBLE,
                             ptMod->LongRunVariance.Val.V_PDOUBLE,
                             ptMod->Sigma.Val.V_PDOUBLE,
                             ptMod->Rho.Val.V_PDOUBLE,
                             &(Met->Res[0].Val.V_DOUBLE));
    }

}

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