Help

```
#include "
href../../../mod/locvolhw1d/locvolhw1d_std/locvolhw1d_std_h_src.pdflocvolhw1d_st
#include "pnl/pnl_cdf.h"
#include "pnl/pnl_finance.h"
#include "pnl/pnl_root.h"
#include "pnl/pnl_cdf.h"
#include "pnl/pnl_finance.h"
#include "pnl/pnl_root.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2012+2) //The "#els
static int CHK_OPT(AP_BGM_Locvolhw)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(AP_BGM_Locvolhw)(void *Opt, void *Mod, PricingMethod *Met)
{
  return AVAILABLE_IN_FULL_PREMIA;
}
#else

static int ApBGMLocvolhw(double S0, NumFunc_1  *p, double T, double csi, double
                         double beta, double rho, double f0t, double *ptprice)
{
  double K;
  double m, nu, B0T, sigma_t, x0, sigma2_black_T, A, sigma1_t, alpha1_T, alpha2_
         d_1, d_2, d_prime_1, d_prime_2, Greek_1, Greek_2, Greek_3, A_1, A_2, A_

  K = p->Par[0].Val.V_PDOUBLE;
  x0 = log(S0);
  m = T * f0t + 0.5 * SQR(csi) * (T + 2 * (exp(-kappa * T) - 1) / kappa - 0.5 *
  nu = SQR(csi) * (T + 2 * exp(-kappa * T) / kappa - 0.5 * exp(-2 * kappa * T) /
  B0T = exp(-m + 0.5 * nu);


  sigma_t = v * exp((beta - 1) * x0); // sigma_t=sigma(t,x0) homogene in time in
  sigma1_t = (beta - 1) * sigma_t; // sigma1_t=d/dx(sigma(t,x))|x=x0 homogene in


  A_1 = T * SQR(sigma_t);
```

```
  A_2 = nu;
  A_3 = -2 * rho * sigma_t *(csi) * ((1 / kappa) * (1 - exp(-kappa * T)) - T) /

  sigma2_black_T = A_1 + A_2 + A_3;

  alpha1_T = (exp(-2 * kappa * T) * sigma_t *sigma1_t / (4 * pow(kappa, 4))) * (


  alpha3_T = (exp(-2 * kappa * T) * sigma_t *sigma1_t) * SQR(rho * csi + exp(kap

  alpha2_T = -alpha1_T - alpha3_T;


  d_1 = (1 / sqrt(sigma2_black_T)) * (log(S0 / (B0T * K)) + 0.5 * sigma2_black_T
  d_2 = d_1 - sqrt(sigma2_black_T);
  d_prime_1 = (1 / sqrt(sigma2_black_T));
  d_prime_2 = d_prime_1;


  A = (S0 / B0T) * pnl_cdfnor(d_1) - K * pnl_cdfnor(d_2);

  Greek_1 = (S0 / B0T) * (pnl_cdfnor(d_1) + d_prime_1 * pnl_normal_density(d_1))

  Greek_2 = (S0 / B0T) * (pnl_cdfnor(d_1) + 2 * d_prime_1 * pnl_normal_density(d

  Greek_3 = (S0 / B0T) * (pnl_cdfnor(d_1) + 3 * d_prime_1 * pnl_normal_density(d


  *ptprice = B0T * (A + alpha1_T * Greek_1 + alpha2_T * Greek_2 + alpha3_T * Gre

  return OK;
}

int CALC(AP_BGM_Locvolhw)(void *Opt, void *Mod, PricingMethod *Met)
{
  TYPEOPT *ptOpt = (TYPEOPT *)Opt;
  TYPEMOD *ptMod = (TYPEMOD *)Mod;
  int status;
  status = ApBGMLocvolhw(ptMod->S0.Val.V_PDOUBLE,
                         ptOpt->PayOff.Val.V_NUMFUNC_1,
                         ptOpt->Maturity.Val.V_DATE - ptMod->T.Val.V_DATE,
```

```
                              ptMod->csi.Val.V_PDOUBLE,
                              ptMod->kappa.Val.V_PDOUBLE,
                              ptMod->v.Val.V_PDOUBLE,
                              ptMod->beta.Val.V_PDOUBLE,
                              ptMod->rho.Val.V_PDOUBLE,
                              ptMod->f0t.Val.V_PDOUBLE,
                              &(Met->Res[0].Val.V_DOUBLE));

  return status;
}

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