Help

```
#include  "
href../../../mod/hw1d/hw1d_vol/hw1d_vol_h_src.pdfhw1d_vol.h"
#include"pnl/pnl_vector.h"
#include"pnl/pnl_random.h"
#include "pnl/pnl_mathtools.h"
#include "pnl/pnl_cdf.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2013+2) //The "#els
static int CHK_OPT(MC_Timer_HW)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(MC_Timer_HW)(void *Opt, void *Mod, PricingMethod *Met)
{
  return AVAILABLE_IN_FULL_PREMIA;
}
#else

//Simulating trajectories of Variance process and the stopping time tau under Hu
void getSpotPathEuler(double a, double nu, double v0,  double V, int generator,
{
  int i, j;
  double Z, x, x1, H, tau;
  double dt;
  double meanV = 0.0;

  dt = V / (double)m;

  for (j = 0; j < n; j++)
    {
      x1 = 2.0 * sqrt(v0) / nu;
      H = dt / x1;
      tau = dt / SQR(x1);
      for (i = 1; i < m; i++)
        {
          Z = pnl_rand_normal(generator);
          x = (sqrt(4.0 * (SQR(x1) +   dt * (4.0 * a / SQR(nu) - 2.0))  + 4.0 *
          H += dt / x;
          tau += dt / SQR(x);
```

```
          x1 = x;
        }
      pnl_mat_set(VT, j, 0, tau * 4.0 / SQR(nu)); // tau
      pnl_mat_set(VT, j, 1, SQR(x)*SQR(nu) / 4.0); // V_tau
      pnl_mat_set(VT, j, 2, H * (2.0 * a / SQR(nu) - 0.5)); //H_\ tau
      meanV += SQR(x) * SQR(nu) / 4.0 / (double)n;

    }
  pnl_mat_set(VT, n - 1, 1, meanV); // V_tau
}

int MCtimerHW(double s0, NumFunc_1 *p, double V, double r, double q, double v0,
{
  int init_mc, simulation_dim;
  PnlMat *VTau;
  double c, d1, d2, vtau, tau, Htau;
  double K, error, mean_price, price;
  int i;


  K = p->Par[0].Val.V_PDOUBLE;
  /* Size of the random vector we need in the simulation */
  simulation_dim = m;

  /* MC sampling */
  init_mc = pnl_rand_init(generator, simulation_dim, n);
  /* Test after initialization for the generator */
  if (init_mc == OK)
    {

      mean_price = error = 0.0;
      VTau = pnl_mat_create(n, 3);

      getSpotPathEuler(a,  nu,   v0,  V, generator,   n,  m,  VTau);

      for (i = 0; i < n; i++)
        {

          tau  = pnl_mat_get(VTau, i, 0);
          vtau = pnl_mat_get(VTau, i, 1);
          Htau = pnl_mat_get(VTau, i, 2);
```

```
            c = 2.0 * rho / nu * (sqrt(vtau) - sqrt(v0)) - rho * Htau - 0.5 * V;

            d1 = (log(s0 / K) + (r - q) * tau + c + (1.0 - SQR(rho)) * V) / sqrt((
            d2 = d1 - sqrt((1.0 - SQR(rho)) * V);

            price = s0 * exp(-q * tau) * exp(c + (1.0 - SQR(rho)) * V / 2.0) * cdf

            mean_price += price;
            error += SQR(price);

          }
        *ptprice = mean_price / (double) n;
        *inf_price = *ptprice - 1.96 * sqrt((error / ((double) n) - (*ptprice) * (
        *sup_price = *ptprice + 1.96 * sqrt((error / ((double) n) - (*ptprice) * (
      }
    pnl_mat_free(&VTau);
    return init_mc;
}

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

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


    return MCtimerHW(ptMod->S0.Val.V_PDOUBLE, ptOpt->PayOff.Val.V_NUMFUNC_1,
                     ptOpt->VarianceBudget.Val.V_PDOUBLE,
                     r,
                     divid, ptMod->Sigma0.Val.V_PDOUBLE,
                     ptMod->Mean.Val.V_PDOUBLE,
                     ptMod->Sigma.Val.V_PDOUBLE,
                     ptMod->Rho.Val.V_PDOUBLE,
                     Met->Par[0].Val.V_LONG,
                     Met->Par[1].Val.V_INT,
                     Met->Par[2].Val.V_ENUM.value,
```

```
                    &(Met->Res[0].Val.V_DOUBLE),
                    &(Met->Res[1].Val.V_DOUBLE),
                    &(Met->Res[2].Val.V_DOUBLE));
}

static int CHK_OPT(MC_Timer_HW)(void *Opt, void *Mod)
{

  if ((strcmp(((Option *)Opt)->Name, "Timer") == 0))
    return OK;

  return  WRONG;
}
#endif //PremiaCurrentVersion

static int MET(Init)(PricingMethod *Met, Option *Opt)
{
  //int type_generator;
  if (Met->init == 0)
    {
      Met->init = 1;
      Met->HelpFilenameHint = "mc_timer_hw";

      Met->Par[0].Val.V_LONG = 15000;
      Met->Par[1].Val.V_INT = 100;
      Met->Par[2].Val.V_ENUM.value = 0;
      Met->Par[2].Val.V_ENUM.members = &PremiaEnumMCRNGs;
    }
  return OK;
}

PricingMethod MET(MC_Timer_HW) =
{
  "MC_Timer_HW",
  { {"N iterations", LONG, {100}, ALLOW},
    {"TimeStepNumber", LONG, {100}, ALLOW},
    {"RandomGenerator", ENUM, {100}, ALLOW},
    {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CALC(MC_Timer_HW),
  { {"Price", DOUBLE, {100}, FORBID},
```

```
    {"Inf Price", DOUBLE, {100}, FORBID},
    {"Sup Price", DOUBLE, {100}, FORBID} ,
    {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CHK_OPT(MC_Timer_HW),
  CHK_mc,
  MET(Init)
};
```