Help

```
#include <stdlib.h>
#include  "
href../../../mod/variancegamma1d/variancegamma1d_std/variancegamma1d_std_h_src.p
#include "pnl/pnl_mathtools.h"
#include "pnl/pnl_complex.h"
#include "pnl/pnl_vector.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2017+2) //The "#els
static int CHK_OPT(AP_COSINEFILTER_vg)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(AP_COSINEFILTER_vg)(void *Opt, void *Mod, PricingMethod *Met)
{
  return AVAILABLE_IN_FULL_PREMIA;
}
#else


static void Valomega (int N, double a, double b, PnlVect *omega)
{
    int j;

    for (j=0;j<N;j++)
    {
        pnl_vect_set(omega,j,((double)j)*M_PI/(b-a));
    }

}

static void Valcf (int N, PnlVect *omega, double theta, double T, double vv, dou
{
    int j;

    for (j=0;j<N;j++)
    {
        double omegaj=pnl_vect_get(omega,j);

        pnl_vect_complex_set(cf,j,Cmul(Cpow_real(RCsub(0.5*sigma*sigma*vv*omegaj
```

```
    }
}

static void cf0 (PnlVectComplex *cf)
{
    pnl_vect_complex_set_real (cf, 0, 0.5*pnl_vect_complex_get_real (cf, 0));
    pnl_vect_complex_set_imag (cf, 0, 0.5*pnl_vect_complex_get_imag (cf, 0));
}

static void VjtM (int N, double a, double b, double K, PnlVect *omega, PnlVect *
{
    int j;
    double omegaj;

    for (j=0;j<N;j++)
    {
        omegaj=pnl_vect_get(omega,j);
        pnl_vect_set(V,j,(-pow((1+pow(omegaj,2)),-1)*(cos((-a)*omegaj)-exp(a)+om
    }
}

static void VjtM0 (double a, double b, double K, PnlVect *V)
{
    pnl_vect_set(V,0,(exp(a)-1.0-a)*(2.0/(b-a))*K);
}

static void sn (double eta,double p,double* ss)
{
    /*Define the filter*/
    double alpha, epsilon;
    epsilon=0.01;
    alpha=-log(epsilon);
    *ss=exp(-alpha*pow(eta,p));
}

static void VecRe (int N, double r, double T,double p, PnlVect *V, PnlVect *omeg
{
    int j;
    double ss;
    double Vj;
```

```
    for (j=0;j<N;j++)
    {
        sn((double)j/(double)N,p,&ss);
        Vj=pnl_vect_get(V,j);
        /*Value with the filter*/
        pnl_vect_set(filt,j,exp(-r*T)*Vj*ss*pnl_vect_complex_get_real (cf,j));
    }
}

static void par (double r, double q, double S0, double T, PnlVect *sum1, double
{
    pnl_vect_set(sum2,0, pnl_vect_get(sum1,0)+S0*exp(-q*T)-K*exp(-r*T));
}
/*////////////////////////////////////////////////////*/
static int cosinefilter_vg(int ifCall, double S0, double sigma, double theta, do
                                       double r, double q,
                                       double T,  double K,
                                       int p,int N,
                                       double *ptprice)
{
  int L;
  double x, a, b, c1, c2, c4,w;
  PnlVect *omega, *V,*fvalue, *filt;
  PnlVectComplex *cf;

  L=10;
  w=1/vv*log(1-theta*vv-sigma*sigma*vv/2);

    omega = pnl_vect_create (N);
    V = pnl_vect_create (N);
    filt = pnl_vect_create (N);
    fvalue = pnl_vect_create (1);
    cf = pnl_vect_complex_create (N);


    /*Transform the stock price to log-asset domain: x=log(S/K)*/
    x=log(S0/K);

    /*Cumulants*/
    c1=((r-q)+theta)*T;
```

```
    c2=(sigma*sigma+vv*theta*theta)*T;
    c4=3*(pow(sigma,4)*vv+2*pow(sigma,4)*pow(vv,3)+4*pow(sigma*theta*vv,2))*T;

    /*Truncation range*/
    a=c1-L*pow(c2+pow(c4,0.5),0.5)+x;
    b=c1+L*pow(c2+pow(c4,0.5),0.5)+x;

    Valomega(N, a, b, omega);

    /*Characteristic function of Variance Gamma model*/
    Valcf(N, omega, theta, T, vv, sigma, x, w, r, q, a, cf);
    cf0(cf);

    /* Fourier Cosine Coefficient of option price at expiry*/
    VjtM(N, a, b, K, omega, V);
    VjtM0(a, b, K, V);

    /* Taking the real part of characteristic function and mulitiply with Fourie
    VecRe(N, r, T, p, V, omega, cf, filt);

    /* Sum up the Fourier Cosine series */
    pnl_vect_set (fvalue, 0, pnl_vect_sum (filt));

    /* The value of a call option is obtained from that of a put option, by put-
    if (ifCall==1)
        par(r, q, S0, T, fvalue, K, fvalue);

    *ptprice = pnl_vect_get(fvalue,0);


    pnl_vect_free(&omega);
    pnl_vect_free(&V);
    pnl_vect_free(&filt);
    pnl_vect_free(&fvalue);
    pnl_vect_complex_free(&cf);

  return OK;
}//============================================================================
int CALC(AP_COSINEFILTER_vg)(void *Opt, void *Mod, PricingMethod *Met)
{
  TYPEOPT *ptOpt = (TYPEOPT *)Opt;
```

```
  TYPEMOD *ptMod = (TYPEMOD *)Mod;
  double r, divid, strike, spot;

  NumFunc_1 *p;
  int res;

  int ifCall;

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

  p = ptOpt->PayOff.Val.V_NUMFUNC_1;
  strike = p->Par[0].Val.V_DOUBLE;
  spot = ptMod->S0.Val.V_DOUBLE;
  ifCall = ((p->Compute) == &Call);



  res =cosinefilter_vg(ifCall, spot, ptMod->Sigma.Val.V_PDOUBLE, ptMod->Theta.Va
                                     r, divid,
   ptOpt->Maturity.Val.V_DATE - ptMod->T.Val.V_DATE,strike, Met->Par[0].Val.V_PI
                                     &(Met->Res[0].Val.V_DOUBLE));

  return res;

}

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

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

  return WRONG;
}


#endif //PremiaCurrentVersion
static int MET(Init)(PricingMethod *Met, Option *Opt)
{
  static int first = 1;
```

```
  if (first)
    {
      Met->HelpFilenameHint = "AP_cosinefilter_vg";
      Met->Par[0].Val.V_PINT = 8;
      Met->Par[1].Val.V_PINT = 128;
      first = 0;
    }

  return OK;
}

PricingMethod MET(AP_COSINEFILTER_vg) =
{
  "AP_CosineFilter_VG",
  { {"Filter order", INT, {100}, ALLOW},
    {"Number of Fourier coefficient", INT, {500}, ALLOW},
    {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CALC(AP_COSINEFILTER_vg),
  { {"Price", DOUBLE, {100}, FORBID},
    {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CHK_OPT(AP_COSINEFILTER_vg),
  CHK_split,
  MET(Init)
};
```