Help

```
/*COS method for European option, Heston model*/
/*Developed by F.Fang, C.W.Oosterlee (2008), implemented by B.Zhang*/

#include <pnl/pnl_mathtools.h>
#include <pnl/pnl_complex.h>
#include <pnl/pnl_vector.h>
# include <pnl/pnl_matrix.h>
# include <pnl/pnl_specfun.h>
# include <pnl/pnl_cdf.h>
#include "
href../../../mod/doublehes1d/doublehes1d_std/doublehes1d_std_h_src.pdfhes1d_std.

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2020+2) //The "#els

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

static void Bound_ab(double L, double S0, double strike, double T, double r, dou
{
    double c1, c2;
    c1=(r-dividend)*T + (1.-exp(-kappa*T))*(theta-v0)/(2.*kappa) - theta*T/2.;
    c2=(sigma*T*kappa*exp(-kappa*T)*(v0-theta)*(8.*kappa*rho-4.*sigma) + kappa*r

    *a  = log(S0/strike) +c1 -L*sqrt(c2);
    *b  = log(S0/strike) +c1 +L*sqrt(c2);
}

static void Cal_omega(double a, double b, double d, int m, int step, int meth_in
{
    int s;
    double hs;
    dcomplex z;
```

```
    pnl_vect_complex_set(omega, 0, Cmul(Complex(0., pow(2.,m)*(meth_index+1)/(b-
      for (s=1 ; s< step; s++)
    {
        hs = ((double)s)*M_PI/step;
        z  = RCmul(d, Cexp(Complex(0., hs)));
        pnl_vect_complex_set(omega, s, Cmul(Complex(0., pow(2.,m)*(meth_index+1)
    }
    pnl_vect_complex_set(omega, step, Cmul(Complex(0., pow(2.,m)*(meth_index+1)/
}

static void charHeston(double a, double b, double d, int m, int step, double S0,
{
    int j;
    double x;
    dcomplex omegaj, D, G,temp1, temp2, temp3, temp4;

    x= log(S0/strike);

    for (j=0;j< step+1;j++)
    {
        omegaj = pnl_vect_complex_get(omega, j);

        D=Cpow_real(Cadd(Cpow_real(RCsub(lambda,Cmul(Complex(0,-rho*eta), omegaj
        G=Cdiv(Csub(RCsub(lambda,Cmul(Complex(0.,-rho*eta),omegaj)),D),Cadd(RCsu

        temp1=RCsub(1,Cexp(CRmul(D,-T)));
        temp2=RCsub(1,Cmul(G,Cexp(CRmul(D,-T))));
        temp3=Csub(RCsub(lambda,Cmul(Complex(0.,-rho*eta),omegaj)),D);
        temp4=RCsub(1,G);

        pnl_vect_complex_set(cf,j,Cmul(Cmul(Cexp(Cadd(Cmul(Complex(0,-(r-q)*T),o
    }
}

static void Cal_Qmj(double a, double b, double d, int m, int step, int meth_inde
{
    int s;
    double hs, temp1;
    dcomplex z, ctemp1, ctemp2, ctemp3;
    temp1 = pow(2., m)*(meth_index+1)/(b-a);
```

```
    pnl_vect_set(Qmj, 0, pow(2., m/2.)*(meth_index+1)/(b-a)*pow(d, -temp1*a)*pow
    for (s=1 ; s< step; s++)
    {
        hs = ((double)s)*M_PI/step;
        z  = RCmul(d, Cexp(Complex(0., hs)));
        ctemp1 = Cpow_real(z, -temp1*a);
        ctemp2 = Cpow_real(Clog(z), meth_index+1);
        ctemp3 = Cpow_real(CRsub(z, 1.), meth_index+1);
        pnl_vect_set(Qmj, s, Creal(Cdiv(Cmul(Cmul(RCmul(pow(2., m/2.)*(meth_inde

    }
    z  = Complex(-d,0.);
    ctemp1 = Cpow_real(z, -temp1*a);
    ctemp2 = Cpow_real(Clog(z), meth_index+1);
    ctemp3 = Cpow_real(CRsub(z, 1.), meth_index+1);
    pnl_vect_set(Qmj, step, Creal(Cdiv(Cmul(Cmul(RCmul(pow(2., m/2.)*(meth_index

}

static void Cal_Cmk(double d, int m, int step, int meth_index, PnlVect *Qmj, Pnl
{
    int k, s;
    double hs, sum;

    sum = pnl_vect_get(Qmj, 0)/2.;
    for (s=1; s<step; s++)
    {
        hs = ((double)s)*M_PI/step;
        sum = sum + pnl_vect_get(Qmj, s);
    }
    sum = sum + pnl_vect_get(Qmj, step)/2.;
    pnl_vect_set(Cmk, 0, sum/step);

    for (k=1; k< (meth_index+1)*(pow(2,m)-1)+1; k++)
    {
        sum = pnl_vect_get(Qmj, 0);
        for (s=1; s<step; s++)
        {
            hs = ((double)s)*M_PI/step;
            sum = sum + 2.*pnl_vect_get(Qmj, s) * cos(k*hs);
        }
```

```
        sum = sum + pow(-1., k)*pnl_vect_get(Qmj, step);
        pnl_vect_set(Cmk, k, sum/step/pow(d, k));
    }
}

static void Cal_vk(int option_index, int meth_index, int m, double a, double b,
{
    double  triangle, beta, gamma, delta, zeta;
    int k;

    triangle = (b-a)/pow(2, m);

    if (meth_index==0)// j=0 for Haar wavelet, else for B-spline with order j
    {
        for (k=0; k<=(meth_index+1)*(pow(2,m)-1); k++)
        {
            beta = a + k*triangle;
            gamma = beta + triangle;
            delta = fmax(0.,beta);
            zeta = fmin(0.,gamma);
            if (option_index == 1) // optiond_index =1 for call option, else for
            {
                if (gamma >0.)
                {
                    pnl_vect_set(vk, k, pow(2., m/2.)*strike*(exp(gamma)-exp(del
                }
                else
                {
                    pnl_vect_set(vk, k, 0.0);
                }
            }
            else
            {
                if (beta <0.)
                {
                    pnl_vect_set(vk, k, pow(2., m/2.)*strike*(exp(beta)-exp(zeta
                }
                else
                {
                    pnl_vect_set(vk, k, 0.0);
                }
```

```
            }
        }
    }
}

static void EuroHestonWavelet(int option_index, int meth_index, int m, int step,
{
    int k;
    double a, b, sum;
    PnlVect *vk, *Qmj, *Cmk;
    PnlVectComplex *cf, *omega;

    omega = pnl_vect_complex_create(step+1);
    cf = pnl_vect_complex_create(step+1);
    Qmj = pnl_vect_create(step+1);
    Cmk = pnl_vect_create( (meth_index+1)*pow(2,m)+1);
    vk  = pnl_vect_create((meth_index+1)*pow(2,m)+1 );
    Bound_ab(L, S0, strike, T, r, q, rho, eta, u, lambda, u0, &a, &b);

    Cal_omega(a, b, d, m, step, meth_index, omega);
    charHeston(a, b, d, m, step, S0, strike, u0, u, r, q, T, eta, rho, lambda, o
    Cal_Qmj(a, b, d, m, step, meth_index, cf, Qmj);
    Cal_Cmk(d, m, step, meth_index, Qmj, Cmk );
    Cal_vk(option_index, meth_index, m, a, b, strike, vk);

    sum = 0.;
    for (k=0; k< (meth_index+1)*(pow(2,m)-1)+1; k++)
    {
        sum = sum + pnl_vect_get(Cmk, k)*pnl_vect_get(vk, k);
    }
    *price = exp(-r*T)*sum;

    pnl_vect_free(&vk);
    pnl_vect_free(&Qmj);
    pnl_vect_free(&Cmk);
    pnl_vect_complex_free(&omega);
    pnl_vect_complex_free(&cf);
}

static int CosineWavelet(double S0, double strike, double T, double r, double di
                  is_call, double *ptprice)
```

```
{
  double L, d;
  int m, step, option_index, meth_index;
  double price;

  if(is_call == TRUE) option_index=1;
  else option_index=0;

    L = 12;
    d = 0.9995;// as defined in page 16//
    m = 7;
    step = pow(2,m);// as defined in page 16//
     meth_index = 0;

    PnlVectComplex *psi;
    psi = pnl_vect_complex_create(m+1);
      if (option_index==1)
    {
        EuroHestonWavelet(option_index, meth_index, m, step, d, L, S0, strike, u
    }
    else
    {
        EuroHestonWavelet(option_index, meth_index, m, step, d, L, S0, strike, u
    }

      *ptprice=price;

    pnl_vect_complex_free(&psi);

  return OK;
}

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

  iscall = FALSE;
  if (ptOpt->PayOff.Val.V_NUMFUNC_1->Compute == &Call) iscall = TRUE;
```

```
  r = log(1. + ptMod->R.Val.V_DOUBLE / 100.);
  divid = log(1. + ptMod->Divid.Val.V_DOUBLE / 100.);
  Met->Res[1].Val.V_DOUBLE = 0.;
  return CosineWavelet(ptMod->S0.Val.V_PDOUBLE,
                ptOpt->PayOff.Val.V_NUMFUNC_1->Par[0].Val.V_PDOUBLE,
                ptOpt->Maturity.Val.V_DATE - ptMod->T.Val.V_DATE, r, divid,
                ptMod->Sigma0.Val.V_PDOUBLE,
                ptMod->LongRunVariance.Val.V_PDOUBLE,
                ptMod->MeanReversion.Val.V_PDOUBLE,
                ptMod->Sigma.Val.V_PDOUBLE,
                ptMod->Rho.Val.V_PDOUBLE,
                iscall,
                &(Met->Res[0].Val.V_DOUBLE));
}

static int CHK_OPT(AP_Cosine_WaveletEuro)(void *Opt, void *Mod)
{
  if ((strcmp(((Option *)Opt)->Name, "CallEuro") == 0) ||
      (strcmp(((Option *)Opt)->Name, "PutEuro") == 0))
    return OK;

  return  WRONG;
}

#endif

static int MET(Init)(PricingMethod *Met, Option *Opt)
{
  if (Met->init == 0)
    {
      Met->Par[0].Val.V_PDOUBLE = 0.1;
      Met->init = 1;
      Met->HelpFilenameHint = "ap_wavelet_cosine_hes1d_euro";
    }
  return OK;
}


PricingMethod MET(AP_Cosine_WaveletEuro) =
{
```

```
  "AP_Cosine_WaveletEuro",
  { {" ", PREMIA_NULLTYPE, {0}, FORBID}},
  CALC(AP_Cosine_WaveletEuro),
  { {"Price", DOUBLE, {100}, FORBID},
    {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CHK_OPT(AP_Cosine_WaveletEuro),
  CHK_ok,
  MET(Init)
};
```