```
   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 "hes1d_std.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
     (2011+2) //The "#else" part of the code will be freely av
    ailable after the (year of creation of this file + 2)

static int CHK_OPT(AP_Cosine_Euro)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(AP_Cosine_Euro)(void *Opt, void *Mod, Pricing
    Method *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 u0,
    double u, double r,
                   double q, double T, double eta, double
```

```
    rho, double
                    lambda, double x, double a, PnlVectCompl
    ex *cf)
{
  int j;

  for (j=0;j<N;j++)
    {
      double omegaj=pnl_vect_get(omega,j);
      dcomplex D, G, temp1, temp2, temp3, temp4;

      D=Cpow_real(Cadd(Cpow_real(RCsub(lambda,Complex(0,rh
    o*eta*omegaj)),2),CRmul(Complex(pow(omegaj,2),omegaj),pow(
    eta,2))),0.5);
      G=Cdiv(Csub(Complex(lambda,-rho*eta*omegaj),D),Cadd(
    Complex(lambda,-rho*eta*omegaj),D));

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

      pnl_vect_complex_set(cf,j,Cmul(Cmul(Cexp(Cadd(
    Complex(0,omegaj*(r-q)*T),CRmul(Cmul(Cdiv(temp1,temp2), temp3),u0
    /pow(eta,2)))), Cexp(RCmul(lambda*u/pow(eta,2), Csub(CRmu
    l(temp3,T),RCmul(2,Clog(Cdiv(temp2,temp4)))))))), Cexp(
    Complex(0,(x-a)*omegaj))));

    }
}

static void cf0 (PnlVectComplex *cf)
{
  pnl_vect_complex_set_real (cf, 0, 0.5*pnl_vect_complex_g
    et_real (cf, 0));
  pnl_vect_complex_set_imag (cf, 0, 0.5*pnl_vect_complex_g
    et_imag (cf, 0));
}

static void VjtM (int N, double a, double b, double K, PnlV
    ect *omega, PnlVect *V)
```

```
{
  int j;

  for (j=0;j<N;j++)
    {
      double omegaj=pnl_vect_get(omega,j);
      pnl_vect_set(V,j,(-pow((1+pow(omegaj,2)),-1)*(cos((-
    a)*omegaj)-exp(a)+omegaj*sin((-a)*omegaj))+pow(omegaj,-1)*
    sin((-a)*omegaj))*(2.0/(b-a))*K);
    }
}

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 VecRe (int N, double r, double T, PnlVect *V,
    PnlVect *omega,
                   PnlVectComplex * cf, PnlVect *fcvec)
{
  int j;

  for (j=0;j<N;j++)
    {
      double Vj=pnl_vect_get(V,j);
      pnl_vect_set(fcvec,j,exp(-r*T)*Vj*pnl_vect_complex_g
    et_real (cf,j));
    }
}

static void par (double r, double q, double S0, double T,
    double K, double *vopt)
{
  *vopt += S0*exp(-q*T)-K*exp(-r*T);
}

static int Cosine(double S0, double K, double T, double r,
    double q, double
                  u0, double u, double lambda, double eta,
```

```
    double rho, int
                  iscall, double *prix)
{
  /* Values of N and L are chosen from the point of view of
     both speed and
   * accuracy. Please do not change them. */

  double x, a, b, c1, c2;
  PnlVect *omega, *V, *fcvec;
  PnlVectComplex *cf;
  int N=128;
  int L=10;

  omega = pnl_vect_create (N);
  V = pnl_vect_create (N);
  fcvec = pnl_vect_create (N);
  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)*T+(1-exp(-lambda*T))*(u-u0)/(2.0*lambda)-0.5*u*
    T;
  c2=(1.0/(8.0*pow(lambda,3)))*(eta*T*lambda*exp(-lambda*T)
    *(u0-u)*(8.0*lambda*rho-4.0*eta)+lambda*rho*eta*(1.0-exp(-
    lambda*T))*(16.0*u-8.0*u0)+2.0*u*lambda*T*(-4.0*lambda*rho*
    eta+pow(eta,2)+4*pow(lambda,2))+pow(eta,2)*((u-2.0*u0)*exp(
    -2*lambda*T)+u*(6.0*exp(-lambda*T)-7)+2*u0)+8*pow(lambda,2
    )*(u0-u)*(1-exp(-lambda*T)));

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

  Valomega(N, a, b, omega);

  /*Characteristic function of Heston model*/
  Valcf(N, omega, u0, u, r, q, T, eta, rho, lambda, x, 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 mu
    litiply with
   * Fourier Cosine Coefficience of option value at expiry*
    /
  VecRe(N, r, T, V, omega, cf, fcvec);

  /* Sum up the Fourier Cosine series */
  *prix =  pnl_vect_sum (fcvec);

  /* The value of a call option is obtained from that of a
    put option, by
   * put-call parity */
  if (iscall == TRUE) par(r, q, S0, T, K, prix);

  pnl_vect_free(&omega);
  pnl_vect_free(&V);
  pnl_vect_free(&fcvec);
  pnl_vect_complex_free(&cf);

  return OK;
}

static int CALC(AP_Cosine_Euro)(void *Opt, void *Mod, Prici
    ngMethod *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) is
    call = 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 Cosine(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.hal.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_Euro)(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_cosine_hes1d_euro";
    }
  return OK;
}


PricingMethod MET(AP_Cosine_Euro)=
```

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

References