```
   Help
extern "C"{
#include "hes1d_vol.h"
}
#include "math/numerics.h"
#include <complex>

#include "math/fft.h"
#include "math/intg.h"

extern "C"{


#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
     (2008+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_HES_REALVAR)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(AP_HES_REALVAR)(void *Opt,void *Mod,PricingMethod
    *Met)
{
  return AVAILABLE_IN_FULL_PREMIA;
}
#else

static complex<double> I(0.0, 1.0);

static complex<double> cphi(double v, double s, double v0,
    double ka, double theta, double sigma, double T);

/*////////////////////////////////////////////////////*/
static int ap_hes_realvar(int ifCall, double v0,double ka,
    double theta,double sigma,double rhow,double r, double divid,
    double T, double Strike, double Spot, double parsigma, double
    parstep, int exp2, double *Price)
{

  double K;
  double shift=parsigma;
```

```
double temp;
long int n;
complex<double> fact;

K=Strike;//p->Par[0].Val.V_DOUBLE;
K=K*K*T/10000.0;

long int Nlimit;
for(n=1,Nlimit=1;n<exp2+1;n++, Nlimit*=2);  //number of
  integral discretization steps
double h=parstep;//step of integrtion

double logstrikestep= 2*M_PI/Nlimit/h; //strike discretiz
  ation step
double A = Nlimit*h/2.0; // integration domain is (-A/2,
  A/2)
double odd=-1.0; // to control Simpson's weights
//expectation of variance
double mval= (theta*T + ( v0 - theta )*( 1.0 - exp(-ka*T)
   )/ka);// /T?

double* y = new double [Nlimit];
double* y_img = new double [Nlimit];
double* k_arr=new double[Nlimit];

double vn = -A;
//double weight = 0.5; //trapezoidal rule weights
double weight = 1./3; //Simpson's rule weights

complex<double> dzeta = exp(-r*T)*(cphi(vn, shift, v0, ka
  , theta, sigma, T))/ ((shift+I*vn)*(shift+I*vn));///2.0;

y[0] = weight*real(dzeta);
y_img[0] = weight*imag(dzeta);
k_arr[0] = K;

//price
for(n=1; n<Nlimit-1; n++){
  vn += h;
  //weight = 1; //trapezoidal rule weights
  odd*= -1.0; //weight = (weight<1) ? 4./3 : 2./3; //Simp
```

```
  son's rule weights
  temp=h*n*K;

  dzeta = exp(-r*T)*exp(I*temp)*(cphi(vn, shift, v0, ka,
  theta, sigma,T) ) / ((shift+I*vn)*(shift+I*vn));
  //price
  y[n] = (1.0+odd*weight)*real(dzeta);
  y_img[n] = (1.0+odd*weight)*imag(dzeta);
  k_arr[n]=K+n*logstrikestep;
}

vn += h;
//weight = 0.5; //trapezoidal rule weights
weight = 1.0/3.0;//Simpson's rule weights

temp=h*n*K;
dzeta = exp(-r*T)*exp(I*temp)*(cphi(vn, shift, v0, ka, th
  eta, sigma,T) ) / ((shift+I*vn)*(shift+I*vn));

y[Nlimit-1] = weight*real(dzeta);
y_img[Nlimit-1] = weight*imag(dzeta);
k_arr[Nlimit-1] = K+(Nlimit-1)*logstrikestep;

fft1d(y,y_img,Nlimit,1);
//*/

if (ifCall)//((p->Compute)==&Call)
  {
    for(n=0;n<Nlimit-1;n++)
      {
        fact=exp((shift-I*A)*k_arr[n])*A/M_PI;
        temp=y[n];
        y[n]=real(fact)*y[n]-imag(fact)*y_img[n] + exp(-
  r*T)*(mval-k_arr[n]);
        y_img[n]=real(fact)*y_img[n]+imag(fact)*temp;
        y[n]=y[n]>0?sqrt(y[n]/T)*100.0:-1;
        k_arr[n]=sqrt(k_arr[n]/T)*100.0;
      }
  }
else
  {
```

```
      for(n=0;n<Nlimit-1;n++)
        {
          fact=exp((shift-I*A)*k_arr[n])*A/M_PI;
          temp=y[n];
          y[n]=real(fact)*y[n]-imag(fact)*y_img[n];
          y_img[n]=real(fact)*y_img[n]+imag(fact)*temp;
          y[n]=y[n]>0?sqrt(y[n]/T)*100.0:-1;
          k_arr[n]=sqrt(k_arr[n]/T)*100.0;
        }
    }

  *Price = y[0];//sqrt(res/T);

  delete [] y;
  delete [] y_img;
  delete [] k_arr;

  return OK;
}

/*-------------------------------------------------------*/
static complex<double> cphi(double v, double s, double v0,
    double ka, double theta, double sigma, double T)
{
  double ss;
  complex<double> d, edt, divedt, aa, bb, val;

  complex<double> x(s, v);

  ss = sigma*sigma;
  d = sqrt(ka*ka + 2.0*ss*x);
  edt = exp(-d*T);
  divedt = 1.0+ka/d +(1.0-ka/d)*edt;
  aa = 2.0*theta*ka/ss*( (ka-d)*T/2.0 + log(2.0) - log(div
    edt) );
  bb = -v0*x/d*2.0*(1.0-edt)/divedt;

  val = exp(aa+bb);

  return val;
```

```
}
/*-----------------------------------------------------------
    -*/


int CALC(AP_HES_REALVAR)(void *Opt,void *Mod,PricingMethod
    *Met)
{
  TYPEOPT* ptOpt=(TYPEOPT*)Opt;
  TYPEMOD* ptMod=(TYPEMOD*)Mod;
  double r, divid, strike, spot;
  NumFunc_1 *p;
  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);

  return ap_hes_realvar(ifCall,
                        ptMod->Sigma0.Val.V_PDOUBLE
                        ,ptMod->MeanReversion.hal.V_PDOUB
    LE,
                        ptMod->LongRunVariance.Val.V_PDOUB
    LE,
                        ptMod->Sigma.Val.V_PDOUBLE,
                        ptMod->Rho.Val.V_PDOUBLE,
                        r,divid,
                        ptOpt->Maturity.Val.V_DATE-ptMod->
    T.Val.V_DATE,
                        strike, spot,
                        Met->Par[0].Val.V_DOUBLE, Met->Par[
    1].Val.V_RGDOUBLE, Met->Par[2].Val.V_INT,
                        &(Met->Res[0].Val.V_DOUBLE)/*PRICE*
    /);

}

static int CHK_OPT(AP_HES_REALVAR)(void *Opt, void *Mod)
```

```
{
  if ((strcmp( ((Option*)Opt)->Name,"CallRealVarEuro")==0 )
    ||strcmp( ((Option*)Opt)->Name,"PutRealVarEuro")==0 )
    return OK;

  return WRONG;
}

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

  if (first)
    {
      Met->Par[0].Val.V_DOUBLE=10.0;
      Met->Par[1].Val.V_RGDOUBLE=0.5;
      Met->Par[2].Val.V_INT=12;

      first=0;
    }

  return OK;
}

PricingMethod MET(AP_HES_REALVAR)=
{
  "AP_HES_REALVAR",
  {   {"Shifting parameter for Laplace transform:", DOUBLE,
    {100},ALLOW    },
      {"Step of discretization for Laplace transform: ", RG
    DOUBLE,{100},ALLOW    },
      {"The log of Nb of points for Laplace transform",
    INT,{10},ALLOW    },
      {" ",PREMIA_NULLTYPE,{0},FORBID}},
  CALC(AP_HES_REALVAR),
  {   {"Price, in annual volatility points", DOUBLE,{100},
    FORBID},
      {" ",PREMIA_NULLTYPE,{0},FORBID}},
  CHK_OPT(AP_HES_REALVAR),
  CHK_ok ,
```

```
  MET(Init)
} ;

/*//////////////////////////////////////*/
}
```

References