```
   Help
extern "C"{
#include "hes1d_vol.h"
#include "numfunc.h"

}

extern "C"{

  int CFPutHeston(double s, double strike, double t,
    double ri, double dividi, double sigma0,double ka,double theta,
    double sigma2,double rhow,double *ptprice, double *ptdelta);
  int CFCallHeston(double s, double strike, double t,
    double ri, double dividi, double sigma0,double ka,double theta,
    double sigma2,double rhow,double *ptprice, double *ptdelta);

#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_VARIANCESWAP)(void *Opt, void *
    Mod)
{
  return NONACTIVE;
}
int CALC(AP_HES_VARIANCESWAP)(void *Opt,void *Mod,Pricing
    Method *Met)
{
return AVAILABLE_IN_FULL_PREMIA;
}
#else

/*///////////////////////////////////////////*/
  static int hes_vanillas(int ifCall, double sigma0,double
    ka,double theta,
                             double sigma2,double rhow,
    double r, double divid,
                              double T, double Strike,
    double Spot, double *price)
  {
    double pprice, pdelta;
    int res;

    if(ifCall)
    {
      res=CFCallHeston(Spot,Strike,T,r,divid,sigma0,ka,thet
    a,sigma2,rhow,&pprice, &pdelta);
    }
    else
    {
      res=CFPutHeston(Spot,Strike,T,r,divid,sigma0,ka,thet
    a,sigma2,rhow,&pprice, &pdelta);

    }
    *price=pprice;
    return res;

  }
  /*//////////////////////////////////////////////////////*/
  static int ap_hes_varswap(  double sigma0,double ka,
    double theta,double sigma2,double rhow,double r, double divid,
    double T, double Strike,
                               double Spot, double *fairv
    al, double *Price)
  {
     double *replStrikes;
 double *replOptions;
 double *replWeights;
 int *CallPuts;
 int flag;
 double S0=Spot;

 double strikestep=0.05*S0, kfirst=0.15*S0;
 double pvfactor=exp(-r*T);

 int k, res, k0, replN=34;
 double optprice, tweight, tprice;


// replication -----------------------------------

 replStrikes = new double[replN];
```

```
replOptions = new double[replN];
replWeights = new double[replN];
CallPuts = new int[replN];

tprice=0.0;

//tstrike=S0;
k=0;
flag=1;

while((k<replN)&&(flag))
{
 replStrikes[k]=kfirst+k*strikestep;
 CallPuts[k]=(S0<=replStrikes[k]);
 flag=!CallPuts[k];
 k++;
}

k0=k-2;
for(;k<replN;k++)
{
 replStrikes[k]=kfirst+k*strikestep;
 CallPuts[k]=1;
}


//weights for puts
tweight=0.0;
//tstrike=replStrikes[k0+1];
for(k=k0;k>=0;k--)
{
 replWeights[k] = /*-(replStrikes[k]-tstrike)*/strikestep/
   (replStrikes[k]*replStrikes[k]);
 tweight+= replWeights[k];
 res=hes_vanillas(CallPuts[k], sigma0,ka,theta, sigma2,rh
   ow,r, divid,T, replStrikes[k],S0*pvfactor, &optprice);

 if(res) {return 1;}
 replOptions[k]=optprice;
 //tstrike = replStrikes[k];
 tprice += replOptions[k]*replWeights[k];
}

//weights for calls
tweight=0;
//tstrike=replStrikes[k0];
for(k=k0+1;k<replN;k++)
{
 replWeights[k] = /*(replStrikes[k]-tstrike)*/strikestep/(
   replStrikes[k]*replStrikes[k]);
 tweight+= replWeights[k];
 res=hes_vanillas(CallPuts[k], sigma0,ka,theta, sigma2,rh
   ow,r, divid,T, replStrikes[k],S0*pvfactor, &optprice);
 if(res) {return 1;}
 replOptions[k]=optprice;
 //tstrike = replStrikes[k];

 tprice+= replOptions[k]*replWeights[k];
}

//portfolio value
tprice*=2.0/T;

//fair strike of variance swap, in annual volatility po
   ints
*fairval= sqrt(tprice/pvfactor)*100.0;
// strike in variance points
kfirst = pvfactor*Strike*Strike;
// price of var swap
*Price= tprice*10000.0-kfirst;

delete [] replStrikes;
delete [] replOptions;
delete [] replWeights;
delete [] CallPuts;


   return OK;
 }
```

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

    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;

    return ap_hes_varswap(
      ptMod->Sigma0.Val.V_PDOUBLE
      ,ptMod->MeanReversion.hal.V_PDOUBLE,
      ptMod->LongRunVariance.Val.V_PDOUBLE,
      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->Res[0].Val.V_DOUBLE)/*FAIRVAL*/,
      &(Met->Res[1].Val.V_DOUBLE)/*PRICE*/);

  }

  static int CHK_OPT(AP_HES_VARIANCESWAP)(void *Opt, void *
    Mod)
  {
    if ((strcmp( ((Option*)Opt)->Name,"VarianceSwap")==0 ))
      return OK;

    return WRONG;
  }

#endif //PremiaCurrentVersion
  static int MET(Init)(PricingMethod *Met,Option *Opt)
  {
    return OK;
  }

  PricingMethod MET(AP_HES_VARIANCESWAP)=
  {
    "AP_HES_VARIANCESWAP", //"Replicating portfolio",
    {   {" ",PREMIA_NULLTYPE,{0},FORBID}},
    CALC(AP_HES_VARIANCESWAP),
    {   {"Fair strike in annual volatility points",DOUBLE,{
    100},FORBID},
        {"Price in 10000 variance points",DOUBLE,{100},FORB
    ID},
        {" ",PREMIA_NULLTYPE,{0},FORBID}},
    CHK_OPT(AP_HES_VARIANCESWAP),
    CHK_ok ,
    MET(Init)
  } ;

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

References