```
   Help
extern "C"{
#include  "hes1d_std.h"
}

#include "math/intg.h"

extern "C"{

static double T,sigma,rho,k, v, r, divid, teta, lambda, S,
    K;

static double charact_funct1(double uu)
{
  double a,b,rs,rsp,sig,tau,tpf1,tpf2, f10, c0, d0;
  dcomplex g,z,w,tp1,tp2,DD,CN,ans,d,expo;


  tau=T;
  a=k*teta;
  rs=rho*sigma;
  rsp=rs*uu;
  sig=sigma*sigma;

  b=k+lambda-rs;
  if(uu==0)
    {
      if(b==0)
  {
    c0=a*T*T/4.0;
          d0=T/2.0;
  }
      else
        {
          c0=0.5*a*(exp(-b*T)+b*T - 1.0)/b/b;
          d0=0.5*(1.0-exp(-b*T))/b;
  }
      f10=log(S/K)+(r-divid)*T+c0+d0*v;

      return f10;
    }
```

```
  z=Complex(-b,rsp);
  z=Cmul(z,z);
  w=RCmul(sig,Complex(-uu*uu,uu));
  d=Csqrt(Csub(z,w));
  tp1=Complex(d.r+b,d.i-rsp);
  tp2=Complex(-d.r+b,-d.i-rsp);
  g=Cdiv(tp2,tp1);

 expo=Cexp(RCmul(-tau,d));
     DD=Csub(Complex(1,0),expo);
     DD=Cdiv(DD,Csub(Complex(1,0),Cmul(g,expo)));
     DD=Cmul(DD,RCmul(1.0/sig,tp2));

  CN=Csub(Cmul(g,expo), Complex(1,0));
  CN=Cdiv(CN,Csub(g, Complex(1,0) ));
  tpf1=a*(tau*tp2.r-2.0*Clog(CN).r)/sig;
  tpf2=a*(tau*tp2.i-2.0*Clog(CN).i)/sig;

  tpf2+=(r-divid)*uu*tau;
  ans=Complex(tpf1+v*DD.r,tpf2+v*DD.i+uu*log(S));
  ans=Cmul(Cexp(ans),Cexp(Complex(0,-uu*log(K))));
  ans=Cdiv(ans,Complex(0,uu));

  return ans.r;
}

static double charact_funct2(double uu)
{
  double a,b,rsp,sig,tau,tpf1,tpf2, f20, c0, d0;
  dcomplex g,z,w,tp1,tp2,DD,CN,ans,d,expo;

  tau=T;
  a=k*teta;
  rsp=rho*sigma*uu;
  sig=sigma*sigma;

  b=k+lambda;
  if(uu==0)
    {
      c0=0.5*a*(exp(-b*T)+b*T - 1.0)/b/b;
      d0=0.5*(1.0-exp(-b*T))/b;
```

```
      f20=log(S/K)+(r-divid)*T-c0-d0*v;

      return f20;
    }
  z=Complex(-b,rsp);
  z=Cmul(z,z);
  w=RCmul(sig,Complex(-uu*uu,-uu));
  d=Csqrt(Csub(z,w));
  tp1=Complex(d.r+b,d.i-rsp);
  tp2=Complex(-d.r+b,-d.i-rsp);
  g=Cdiv(tp2,tp1);
  expo=Cexp(RCmul(-tau,d));
  DD=Csub(Complex(1,0),expo);
  DD=Cdiv(DD,Csub(Complex(1,0),Cmul(g,expo)));
  DD=Cmul(DD,RCmul(1.0/sig,tp2));

  CN=Csub(Cmul(g,expo), Complex(1,0));
  CN=Cdiv(CN,Csub(g, Complex(1,0) ));
  tpf1=a*(tau*tp2.r-2.0*Clog(CN).r)/sig;
  tpf2=a*(tau*tp2.i-2.0*Clog(CN).i)/sig;

  tpf2+=(r-divid)*uu*tau;
  ans=Complex(tpf1+v*DD.r,tpf2+v*DD.i+uu*log(S));
  ans=Cmul(Cexp(ans),Cexp(Complex(0,-uu*log(K))));
  ans=Cdiv(ans,Complex(0,uu));

  return ans.r;
}

static double  probabilities(int n)
{
  double tp, cinf, f0, Lamb, abserr;

  cinf=sqrt(1.0-rho*rho)/sigma*(v+k*teta*T);

  if(n==1)
    {
      f0=charact_funct1(0.0);

      Lamb=2.0*(log(fabs(f0))+12.0*log(10.0))/cinf;
```

```
      intg(0.0, Lamb, charact_funct1, 1e-14, 1e-10, &tp, &
    abserr);

      tp=0.5+tp/M_PI;
      return tp;

    }
  else
    {
      f0=charact_funct2(0.0);

      Lamb=2.0*(log(fabs(f0))+12.0*log(10.0))/cinf;

      intg(0.0, Lamb, charact_funct2, 1e-14, 1e-10, &tp, &
    abserr);

      tp=0.5+tp/M_PI;
      return tp;
    }
}

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)
{
  double proba1,proba2,temp;

  K=strike;
  S=s;
  T=t;
  sigma=sigma2;
  v=sigma0;
  teta=theta;
  lambda=0.;
  r=ri;
  divid=dividi;
  rho=rhow;
  k=ka;

  proba1=probabilities(1);
  proba2=probabilities(2);
```

```
  temp=s*proba1*exp(-divid*t);
  temp-=K*exp(-r*t)*proba2;

  /* Price*/
  *ptprice=temp;

  /* Delta */
  *ptdelta=proba1*exp(-divid*t);

  return OK;
}

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

  if(ptMod->Sigma.Val.V_PDOUBLE==0.0)
    {
      Fprintf(TOSCREEN,"BLACK-SHOLES MODEL{n{n{n");
      return WRONG;
    }
  else
    {
      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;

      return CFCallHeston(ptMod->S0.Val.V_PDOUBLE,
        strike/*ptOpt->PayOff.Val.V_NUMFUNC_1*/,
        ptOpt->Maturity.Val.V_DATE-ptMod->T.Val.V_DATE,
        r,
        divid, 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,
        &(Met->Res[0].Val.V_DOUBLE),
        &(Met->Res[1].Val.V_DOUBLE)
        );
    }

}



static int CHK_OPT(CF_CallHeston)(void *Opt, void *Mod)
{
  return strcmp( ((Option*)Opt)->Name,"CallEuro");
}



static int MET(Init)(PricingMethod *Met,Option *Opt)
{
  if ( Met->init == 0)
    {
      Met->init=1;
    }

  return OK;
}

PricingMethod MET(CF_CallHeston)=
{
  "CF_Call_Heston",
  {{" ",PREMIA_NULLTYPE,{0},FORBID}},
  CALC(CF_CallHeston),
  {{"Price",DOUBLE,{100},FORBID},
   {"Delta",DOUBLE,{100},FORBID} ,
   {" ",PREMIA_NULLTYPE,{0},FORBID}},
  CHK_OPT(CF_CallHeston),
  CHK_ok,
  MET(Init)
};

}
```

References