Help

```
#include  "scott1d_std.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
     (2010+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_Alos_Scott)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(AP_Alos_Scott)(void*Opt,void *Mod,PricingMethod *
    Met)
{
  return AVAILABLE_IN_FULL_PREMIA;
}
#else

static double d1(double x,double t,double s,double K,
    double r,double T){
  double d=(log(x/K)+(r+s*s/2)*(T-t))/(s*sqrt(T-t));
  return d;
}
static double H(double t, double x, double v,double K,
    double r,double T){
  double a,d,HH;
  a=d1(x,t,v,K,r,T)*d1(x,t,v,K,r,T);
  d=v*sqrt((T-t)*2*M_PI);
  HH=exp(-a/2)/d*x*(1-d1(x,t,v,K,r,T)/v/sqrt(T));
  return HH;
}

static double diffH(double v,double T, double S, double K,
    double r){
  return(-0.5/pow(v,3)*pow(2,0.5)/pow(M_PI,0.5)/pow(T,0.5)
    *(log(S/K)+(r+1./2*v*v)*T)/T/S*exp(-1./2*pow((log(S/K)+(r+
    1./2*v*v)*T),2)/pow(v,2)/T)*(1-1./2*(log(S/K)+(r+1./2*v*v)
    *T)/pow(v,2)/pow(T,0.5)*pow(2,0.5)/sqrt(M_PI)/pow(T,0.5))-
    1./2/pow(v,3)/M_PI/pow(T,3./2)*exp(-1./2*pow(log(S/K)+(r+1
    ./2*v*v)*T,2)/pow(v,2)/T)/S);
}
```

```
static double g(double u,double s,double sigma0, double ka
    ppa,double theta,double lambda){
  return exp(2*(theta+(log(sigma0)-theta)*exp(-kappa*u))+(
    theta+(log(sigma0)-theta)*exp(-kappa*s))+pow(lambda,2)*(1-
    exp(2*kappa*u))+lambda*lambda*exp(-kappa*(u+s))*(exp(2*kapp
    a*u)-1)+pow(lambda,2)*(1-exp(2*kappa*u))*1./2-kappa*(u-s));
}

static double h(double s,double theta, double sigma0,
    double alpha, double lambda){
  return exp(2*(theta+(log(sigma0)-theta)*exp(-alpha*s))+
    pow(lambda,2)*(1-exp(2*alpha*s)));
}

int ApAlosScott(double S,NumFunc_1  *p, double T, double r,
     double divid, double v0,double kappa,double theta,double
    sigma,double rho,double *ptprice, double *ptdelta)
{
  int flag_call,j,k,N;
  double K,prix,delta,price_bs,delta_bs;
  double I,Ij,NVol,vol,lambda,d,sj,rk;
  double sigma0;

  K=p->Par[0].Val.V_PDOUBLE;
  sigma0=v0;
  if ((p->Compute)==&Call)
    flag_call=1;
  else
    flag_call=0;;

  I=0;
  NVol=0;
  lambda=sigma/sqrt(kappa);
  N=1000;
  for(j=0;j<N;j++){
    Ij=0;
    sj=(double)j*T/(double)N;
    NVol+=T/N*h(sj,theta,sigma0,kappa,lambda);
    //Computation of the quantity denote by v0* in the pap
    er by the approximation of Riemman
    for(k=0;k<N;k++){
      rk= sj+(T-sj)*(double)k/N;
      Ij+=g(rk,sj,sigma0,kappa,theta,lambda)*(T-sj)/N;
    }
    I+=T/N*Ij;
    //Calculation of the quantity denote by I in the paper
    by the approximation method of Riemman
  }
  vol=sqrt(NVol*1/T);

  if(flag_call==1){
    pnl_cf_call_bs(S,K,T,r,divid,vol,&price_bs,&delta_bs);
  prix=price_bs+sigma*rho*H(0,S,vol,K,r,T)*I;
    d=diffH(vol,T,S,K,r);
  delta=delta_bs+sigma*rho*I*d;
  }
  else{
    pnl_cf_put_bs(S,K,T,r,divid,vol,&price_bs,&delta_bs);
  prix=price_bs+sigma*rho*H(0,S,vol,K,r,T)*I;
    d=diffH(vol,T,S,K,r);
  delta=delta_bs+sigma*rho*I*d;
  }

  /* Price*/
  *ptprice=prix;

  /* Delta */
  *ptdelta=delta;

  return OK;
}

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

  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.);
      //strike=p->Par[0].Val.V_DOUBLE;

      return ApAlosScott(ptMod->S0.Val.V_PDOUBLE,
        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(AP_Alos_Scott)(void *Opt, void *Mod)
{
  if ((strcmp( ((Option*)Opt)->Name,"CallEuro")==0)
      ||(strcmp( ((Option*)Opt)->Name,"PutEuro")==0))

    return OK;
  return WRONG;
}

#endif //PremiaCurrentVersion
static int MET(Init)(PricingMethod *Met,Option *Opt)
{
  if ( Met->init == 0)
    {
      Met->init=1;
    }
  return OK;
}

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

References