Help

```
#ifndef  _BS1D_LIM_H
#define _BS1D_LIM_H

#include "bs1d.h"
#include "lim.h"

#include "mathtools.h"
#include "random.h"
#include "numfunc.h"
#include "transopt.h"

static int formula(double s,double k,double r,
    double divid,double sigma,double t,double l, double re
    bate,int phi,int eta,double *A,double *B,double *
    C,double *D,double *E,double *F,double *dA,
    double *dB,double *dC,double *dD,double *dE,double *dF
    )
    {
    double b,x1,x2,y1,y2,z,mu,lambda,sigmasqrt;

    sigmasqrt=sigma*sqrt(t);
    b=r-divid;
    mu=(b-SQR(sigma)/2.)/SQR(sigma);
    lambda=sqrt(SQR(mu)+2.*r/SQR(sigma));
    x1=log(s/k)/sigmasqrt + (1+mu)*sigmasqrt;
    x2=log(s/l)/sigmasqrt + (1+mu)*sigmasqrt;
    y1=log(SQR(l)/(s*k))/sigmasqrt+(1+mu)*sigmasq
    rt;
    y2=log(l/s)/sigmasqrt + (1+mu)*sigmasqrt;
    z=log(l/s)/sigmasqrt + lambda*sigmasqrt;
    *A=phi*s*exp((b-r)*t)*N(phi*x1)-phi*k*exp(-r*
    t)*N(phi*x1-phi*sigmasqrt);
    *B=phi*s*exp((b-r)*t)*N(phi*x2)-phi*k*exp(-r*
    t)*N(phi*x2-phi*sigmasqrt);
    *C=phi*s*exp((b-r)*t)*pow(l/s,2.*(1.+mu))*N(
    eta*y1)-
        phi*k*exp(-r*t)*pow(l/s,2.*mu)*N(eta*y1-
    eta*sigmasqrt);
    *D=phi*s*exp((b-r)*t)*pow(l/s,2.*(1.+mu))*N(
    eta*y2)-
```

```
    phi*k*exp(-r*t)*pow(l/s,2.*mu)*N(eta*y2-
eta*sigmasqrt);
*E=rebate*exp(-r*t)*(N(eta*x2-eta*sigmasqrt)-
pow(l/s,2.*mu)*N(eta*y2-eta*sigmasqrt));
*F=rebate*(pow(l/s,mu+lambda)*N(eta*z)+pow(l/
s,mu-lambda)*N(eta*z-2.*eta*lambda*sigmasqrt));

*dA=phi*exp(-divid*t)*N(phi*x1);

*dB=phi*exp(-divid*t)*N(phi*x2)+exp(-divid*t)
*nd(x2)/(sigma*sqrt(t))*(1.-k/l);

*dC=-phi*2.*mu*pow(l/s,2.*mu)*(1./s)*(s*exp(-
divid*t)*SQR(l/s)*N(eta*y1)
    -k*exp(-r*t)*N(eta*y1-eta*sigma*sqrt(t)))
-
    phi*pow(l/s,2.*mu+2.)*exp(-divid*t)*N(et
a*y1);

*dD=-2.*mu*(phi/s)*pow(l/s,2.*mu)*(s*exp(-div
id*t)*SQR(l/s)*N(eta*y2)-
    k*exp(-r*t)*N(eta*(y2-sigma*sqrt(t))))-
    phi*pow(l/s,2.*mu+2.)*exp(-divid*t)*N(et
a*y2)-phi*eta*exp(-divid*t)*
    pow(l/s,2.*mu+2.)*nd(y2)/(sigma*sqrt(t))*
(1.-k/l);

*dE=2.*(rebate/s)*exp(-r*t)*pow(l/s,2.*mu)*(
N(eta*(y2-sigma*sqrt(t)))*mu+
    eta*nd(y2-sigma*sqrt(t))/(sigma*sqrt(t)))
;

*dF=-pow(l/s,mu+lambda)*(rebate/s)*((mu+lamb
da)*N(eta*z)+(mu-lambda)*pow(s/l,2.*lambda)*N(et
a*(z-2.*lambda*sigma*sqrt(t))))-
    2.*eta*rebate*pow(l/s,mu+lambda)*nd(z)/(
s*sigma*sqrt(t));

return OK;
}
```

```
static double Boundary(double s,NumFunc_1*p,
    double t,double r,double divid,double sigma)
{
  int dummy;
  double price,delta;

  if  ((p->Compute)==&Call)
    dummy=Call_BlackScholes_73(s,p->Par[0].Val.V_
    PDOUBLE,t,r,divid,sigma,&price,&delta);
  else if ((p->Compute)==&Put)
    dummy=Put_BlackScholes_73(s,p->Par[0].Val.V_
    PDOUBLE,t,r,divid,sigma,&price,&delta);

  return price;
}

#endif
```

References