Help

```
#ifndef  _BS1D_LIM_H
#define _BS1D_LIM_H

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

#include "pnl/pnl_mathtools.h"
#include "pnl/pnl_random.h"
#include "pnl/pnl_cdf.h"
#include "numfunc.h"
#include "transopt.h"

#ifdef WITH_formula
static int formula(double s,double k,double r,double divid,
    double sigma,double t,double l, double rebate,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)*sigmasqrt;
  y2=log(l/s)/sigmasqrt + (1+mu)*sigmasqrt;
  z=log(l/s)/sigmasqrt + lambda*sigmasqrt;
  *A=phi*s*exp((b-r)*t)*cdf_nor(phi*x1)-phi*k*exp(-r*t)*cdf
    _nor(phi*x1-phi*sigmasqrt);
  *B=phi*s*exp((b-r)*t)*cdf_nor(phi*x2)-phi*k*exp(-r*t)*cdf
    _nor(phi*x2-phi*sigmasqrt);
  *C=phi*s*exp((b-r)*t)*pow(l/s,2.*(1.+mu))*cdf_nor(eta*y1)
    -
    phi*k*exp(-r*t)*pow(l/s,2.*mu)*cdf_nor(eta*y1-eta*sigma
    sqrt);
  *D=phi*s*exp((b-r)*t)*pow(l/s,2.*(1.+mu))*cdf_nor(eta*y2)
    -
```

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

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

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

  *dC=-phi*2.*mu*pow(l/s,2.*mu)*(1./s)*(s*exp(-divid*t)*SQ
    R(l/s)*cdf_nor(eta*y1)
          -k*exp(-r*t)*cdf_nor(eta*y1-eta*sigma*sqrt(
    t)))-
    phi*pow(l/s,2.*mu+2.)*exp(-divid*t)*cdf_nor(eta*y1);

  *dD=-2.*mu*(phi/s)*pow(l/s,2.*mu)*(s*exp(-divid*t)*SQR(l/
    s)*cdf_nor(eta*y2)-
             k*exp(-r*t)*cdf_nor(eta*(y2-sigma*sqrt(t))
    ))-
    phi*pow(l/s,2.*mu+2.)*exp(-divid*t)*cdf_nor(eta*y2)-ph
    i*eta*exp(-divid*t)*
    pow(l/s,2.*mu+2.)*pnl_normal_density(y2)/(sigma*sqrt(t)
    )*(1.-k/l);

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

  *dF=-pow(l/s,mu+lambda)*(rebate/s)*((mu+lambda)*cdf_nor(
    eta*z)+(mu-lambda)*pow(s/l,2.*lambda)*cdf_nor(eta*(z-2.*lam
    bda*sigma*sqrt(t))))-
    2.*eta*rebate*pow(l/s,mu+lambda)*pnl_normal_density(z)/
    (s*sigma*sqrt(t));

  return OK;
}
#endif
```

```
#ifdef WITH_boundary
static double Boundary(double s,NumFunc_1*p,double t,
    double r,double divid,double sigma)
{
  double price,delta;

  if  ((p->Compute)==&Call)
    pnl_cf_call_bs(s,p->Par[0].Val.V_PDOUBLE,t,r,divid,si
    gma,&price,&delta);
  else if ((p->Compute)==&Put)
   pnl_cf_put_bs(s,p->Par[0].Val.V_PDOUBLE,t,r,divid,sigma,
    &price,&delta);

  return price;
}
#endif

#endif
```

References