Help

```
#ifndef  _BS1D_DOUBLIM_H
#define _BS1D_DOUBLIM_H

#include "bs1d.h"
#include "doublim.h"

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

static int PutOut_KunitomoIkeda_91(double s,
    NumFunc_1  *L,NumFunc_1  *U,NumFunc_1  *Rebate,NumFunc_
    1 *PayOff,double t,double r,double divid,double
    sigma,double *ptprice,double *ptdelta)
    {
    double d1,d2,d3,d4,mu1,mu2,mu3,u,l,sum1,sum2,
    time,k,delta1,delta2,E;
    int n;

    sum1=0.0;
    sum2=0.0;
    time=0.;
    u=(U->Compute)(U->Par,time);
    l=(L->Compute)(L->Par,time);
    k= PayOff->Par[0].Val.V_PDOUBLE;
    delta1=0.0;
    delta2=0.0;
    E=l*exp(delta2*t);

    for(n=-5;n<=5;n++)
    {
        mu1=2.*(r-divid-delta2-(double)n*(delta1-
    delta2))/SQR(sigma)+1.0;
        mu2=2.*(double)n*(delta1-delta2)/SQR(sig
    ma);
        mu3=mu1;
        d1=(log(s*pow(u,2.0*(double)n)/(E*pow(l,2
```

```
    .0*(double)n)))+
            (r-divid+SQR(sigma)/2.0)*t)/(sigma*sq
    rt(t));
        d2=(log(s*pow(u,2.0*n)/(k*pow(l,2.0*(
    double)n)))+
            (r-divid+SQR(sigma)/2.0)*t)/(sigma*sq
    rt(t));
        d3=(log(pow(l,2.0*(double)n+2.)/(E*s*pow(
    u,2.0*(double)n)))+
            (r-divid+SQR(sigma)/2.0)*t)/(sigma*sq
    rt(t));
        d4=(log(pow(l,2.0*(double)n+2.)/(k*s*pow(
    u,2.0*(double)n)))+
            (r-divid+SQR(sigma)/2.0)*t)/(sigma*sq
    rt(t));
        sum2+=pow(pow(u/l,(double)n),mu1-2.)*pow(
    l/s,mu2)*(N(d1-sigma*sqrt(t))-
            N(d2-sigma*sqrt(t)))-
            pow(pow(l,(double)(n+1))/(s*pow(u,(
    double)n)),mu3-2.)
      *(N(d3-sigma*sqrt(t))-N(d4-sigma*sqrt(t))
     );
        sum1+=pow(pow(u/l,(double)n),mu1)*pow(l/
    s,mu2)*(N(d1)-N(d2))-
            pow(pow(l,(double)(n+1))/(s*pow(u,(
    double)n)),mu3)*(N(d3)-N(d4));
        }

    /*Price*/
    *ptprice=k*exp(-r*t)*sum2-s*exp(-divid*t)*su
    m1;

    /*Delta*/
    *ptdelta=0.0;

    return OK;
    }


static int CallOut_KunitomoIkeda_91(double s,
    NumFunc_1 *L,NumFunc_1 *U,NumFunc_1  *Rebate,NumFunc_1
```

```
*PayOff,double t,double r,double divid,double si
gma,double *ptprice,double *ptdelta)
{
double d1,d2,d3,d4,mu1,mu2,mu3,F,u,l,sum1,su
m2,time,k,delta1,delta2;
int n;


sum1=0.0;
sum2=0.0;
time=0.;
u=(U->Compute)(U->Par,time);
l=(L->Compute)(L->Par,time);
k= PayOff->Par[0].Val.V_PDOUBLE;
delta1=0.0;
delta2=0.0;
F=u*exp(delta1*t);
for(n=-5;n<=5;n++)
{
    mu1=2.*(r-divid-delta2-(double)n*(delta1-
delta2))/SQR(sigma)+1.0;
    mu2=2.*(double)n*(delta1-delta2)/SQR(sig
ma);
    mu3=mu1;
    d1=(log(s*pow(u,2.0*(double)n)/(k*pow(l,2
.0*(double)n)))+
        (r-divid+SQR(sigma)/2.0)*t)/(sigma*sq
rt(t));
    d2=(log(s*pow(u,2.0*n)/(F*pow(l,2.0*(
double)n)))+
        (r-divid+SQR(sigma)/2.0)*t)/(sigma*sq
rt(t));
    d3=(log(pow(l,2.0*(double)n+2.)/(k*s*pow(
u,2.0*(double)n)))+
        (r-divid+SQR(sigma)/2.0)*t)/(sigma*sq
rt(t));
    d4=(log(pow(l,2.0*(double)n+2.)/(F*s*pow(
u,2.0*(double)n)))+
        (r-divid+SQR(sigma)/2.0)*t)/(sigma*sq
rt(t));
    sum2+=pow(pow(u/l,(double)n),mu1-2.)*pow(
```

```
    l/s,mu2)*(N(d1-sigma*sqrt(t))-
            N(d2-sigma*sqrt(t)))-
            pow(pow(l,(double)(n+1))/(s*pow(u,(
    double)n)),mu3-2.)*(N(d3-sigma*sqrt(t))-N(d4-sigma*sq
    rt(t)));
        sum1+=pow(pow(u/l,(double)n),mu1)*pow(l/
    s,mu2)*(N(d1)-N(d2))-
          pow(pow(l,(double)(n+1))/(s*pow(u,(
    double)n)),mu3)*(N(d3)-N(d4));
     }

    /*Price*/
    *ptprice=s*exp(-divid*t)*sum1-k*exp(-r*t)*su
    m2;

    /*Delta*/
    *ptdelta=0.;

    return OK;
    }

static double Boundary(double s,NumFunc_1*p,
    double t,double r,double divid,double sigma)
{
  int dummy;
  double price=0.,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