Help

```
/* We need Nd1 here */
#define USE_ND1 1
#include  "bs1d_std.h"
#define INC 1.0e-5 /*Relative Increment for Delta-Hedging*/

/*Put Whaley Exponent*/
static double WhaleyPut_Exp(double r,double divid,double si
    gma,double T)
{
  double ratio = 2.0 * (r-divid) / (sigma * sigma);
  double delta = (ratio - 1.0);

  if(r==0.)
    delta=SQR(delta)+4.0*(2.0/(sigma*sigma))/T;
  else
    delta=SQR(delta)+4.0*(2.0*r/(sigma*sigma))/(1.0-exp(-r*
    T));

  return 0.5*(1.-ratio-sqrt(delta));
}

/*Call Whaley Exponent*/
static double WhaleyCall_Exp(double r,double divid,double
    sigma,double T)
{
  double ratio = 2.0 * (r-divid) / (sigma * sigma);
  double delta = (ratio - 1.0);

  if(r==0)
    delta=SQR(delta)+4.0*(2.0/(sigma*sigma))/T;
  delta=SQR(delta)+4.0*(2.0*r/(sigma*sigma))/(1.0-exp(-r*T)
    );
  return 0.5*(1.-ratio+sqrt(delta));
}

/*Put Critical Price*/
static double Contact_PointPut(double r,double divid,
    double sigma,double T,double K,double (*exponent_method)(double,
    double,double,double))
{
  const double precision = 0.00001;
  double previous;
  double exponent = (*exponent_method)(r,divid,sigma,T);
  double current  = K;
  double put_price,put_delta;

  do {
    previous = current;
    pnl_cf_put_bs(previous,K,T,r,divid,sigma,&put_price,&
    put_delta);
    current=-exponent*(K-put_price)/((1.-exp(-divid*T)*Nd1(
    previous,r,divid,-sigma,T,K))-exponent);
  } while(!(fabs((previous-current)/current)<=precision));

  return current;
}

/*Call Critical Price*/
static double Contact_PointCall(double r,double divid,
    double sigma,double T,double K,
                                double (*exponent_method)(
    double,double,double,double))
{
  const double precision = 0.00001;
  double previous;
  double exponent = (*exponent_method)(r,divid,sigma,T);
  double current=K;
  double call_price,call_delta;

  do {
    previous =current;
    pnl_cf_call_bs(previous,K,T,r,divid,sigma,&call_price,&
    call_delta);
    current=exponent*(K+call_price)/(-(1.-exp(-divid*T)*Nd1
    (previous,r,divid,sigma,T,K))+exponent);
  } while(!(fabs((previous-current)/current)<=precision));

  return current;
}

/*Whaley Formula*/
```

```
static double Formula_Whaley(double r,double divid,double
    sigma,double T,double x,double K,NumFunc_1 *p)
{

  double exponent;
  double critical_price;
  double a,put_price,put_delta,call_price,call_delta;

  if ((p->Compute)==&Put)
    {
      critical_price=Contact_PointPut(r,divid,sigma,T,K,Wh
    aleyPut_Exp);
      if(x < critical_price)
        {
    return (K - x);
        }
      else
        {
    exponent=WhaleyPut_Exp(r,divid,sigma,T);
    a=critical_price*(1.-exp(-divid*T)*Nd1(critical_price,
    r,divid,-sigma,T,K))/(-exponent);
    pnl_cf_put_bs(x,K,T,r,divid,sigma,&put_price,&put_delt
    a);
    return put_price+a*pow(x/critical_price,exponent);
        }
    }
  else
    if ((p->Compute)==&Call)
      {
  critical_price=Contact_PointCall(r,divid,sigma,T,K,Whale
    yCall_Exp);
  if(x >= critical_price)
    {
      return (x - K);
    }
  else
    {
      exponent=WhaleyCall_Exp(r,divid,sigma,T);
      a=critical_price*(1.-exp(-divid*T)*Nd1(critical_
    price,r,divid,sigma,T,K))/exponent;
      pnl_cf_call_bs(x,K,T,r,divid,sigma,&call_price,&
    call_delta);
      return call_price+a*pow(x/critical_price,exponent);
    }
      }

  /*Never reached normally*/
  return 0.;
}

static int Whaley_81(double s,NumFunc_1 *p,double t,double
    r,double divid,double sigma,double *ptprice,double *ptdelt
    a){
  double s_plus,s_minus;

  s_plus=s*(1.+INC);
  s_minus=s*(1.-INC);

  /*Price*/
  *ptprice=Formula_Whaley(r,divid,sigma,t,s,p->Par[0].Val.
    V_DOUBLE,p);

  /*Delta*/
  *ptdelta=(Formula_Whaley(r,divid,sigma,t,s_plus,p->Par[0]
    .Val.V_DOUBLE,p)-Formula_Whaley(r,divid,sigma,t,s_minus,p-
    >Par[0].Val.V_DOUBLE,p))/(2.*s*INC);

  return OK;
}

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

  r=log(1.+ptMod->R.Val.V_DOUBLE/100.);
  divid=log(1.+ptMod->Divid.Val.V_DOUBLE/100.);

  return Whaley_81(ptMod->S0.Val.V_PDOUBLE,
       ptOpt->PayOff.Val.V_NUMFUNC_1,ptOpt->Maturity.Val.
    V_DATE-ptMod->T.Val.V_DATE,
       r,divid,ptMod->Sigma.Val.V_PDOUBLE,
       &(Met->Res[0].Val.V_DOUBLE),&(Met->Res[1].Val.V_
    DOUBLE));
}


static int CHK_OPT(AP_Whaley)(void *Opt, void *Mod)
{
  if ( (strcmp( ((Option*)Opt)->Name,"CallAmer")==0) || (
    strcmp( ((Option*)Opt)->Name,"PutAmer")==0) )
    return OK;

  return WRONG;
}

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

  return OK;
}

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

References