```
   Help
#include  "bs1d_std.h"

/*Critical Price*/
static double PutCriticalPrice(double r,double divid,
    double sigma,double T,double K)
{
  const double precision = 0.0001;
  double previous;
  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=K-put_price;
  } while(!(fabs((previous-current)/current)<=precision));
  return current;
}

static double CallCriticalPrice(double r,double divid,
    double sigma,double T,double K)
{
  const double precision = 0.0001;
  double previous;
  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=K+call_price;
  } while(!(fabs((previous-current)/current)<=precision));
  return current;
}

/* 2-points Ho-Stapleton-Subrahmanyam AP*/
static int HoStapletonSubrahmanyam_94(double s,NumFunc_1*p,
    double t,double r,double divid,double sigma,double *ptprice,
    double *ptdelta)
```

```
{
  double p1,p2,crit12,k,price,delta,val,w1,w2,d1,d2,d1c,d2
    c;

  k=p->Par[0].Val.V_PDOUBLE;

  if ((p->Compute)==&Call)
    {
      val=-1.;
      pnl_cf_call_bs(s,k,t,r,divid,sigma,&price,&delta);
      p1=price;
      crit12=CallCriticalPrice(r,divid,sigma,t/2.,k);
    }
  else
    {
      val=1.;
      pnl_cf_put_bs(s,k,t,r,divid,sigma,&price,&delta);
      p1=price;
      crit12=PutCriticalPrice(r,divid,sigma,t/2.,k);
    }
  d1c= (log(s/crit12)+(r-divid+0.5*SQR(sigma))*t/2.)/(sigma
    *sqrt(t/2.));
  d2c=d1c-sigma*sqrt(t/2.);
  d1=(log(s/k)+(r-divid+0.5*SQR(sigma))*t)/(sigma*sqrt(t));
  d2=d1-sigma*sqrt(t);
  w1=exp(-divid*t/2.)*cdf_nor(-val*d1c)+exp(-divid*t)*pnl_
    cdf2nor(val*d1c,-val*d1,-sqrt(0.5));
  w2=exp(-r*t/2.)*cdf_nor(-val*d2c)+exp(-r*t)*pnl_cdf2nor(
    val*d2c,-val*d2,-sqrt(0.5));
  p2=val*(k*w2-s*w1);

  /*Price*/
  *ptprice=SQR(p2)/p1;

  /*Delta*/
  *ptdelta=(*ptprice)*(-2.*val*w1/p2-delta/p1);

  return OK;
}
```

```
int CALC(AP_HoStapletonSubrahmanyam)(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 HoStapletonSubrahmanyam_94(ptMod->S0.Val.V_PDOUB
    LE,
            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_HoStapletonSubrahmanyam)(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;
    }

  return OK;
}

PricingMethod MET(AP_HoStapletonSubrahmanyam)=
```

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

References