

Help

```

#include "bs1d_std.h"
#include "error_msg.h"
#define INC 1.0e-5 /*Relative Increment for Delta-Hedging*/

/*Phi function*/
static double phi(double r,double divid,double sigma,
    double s,double t,double gamma,double H,double I)
{
    double res,lambda,d,k,b;

    b=r-divid;
    lambda=(-r+gamma*b+0.5*gamma*(gamma-1.)*SQR(sigma))*t;
    d=-((log(s/H)+(b+(gamma-0.5)*SQR(sigma))*t)/(sigma*sqrt(t))
    );
    k=2*b/SQR(sigma)+(2.*gamma-1.);
    res=exp(lambda)*pow(s,gamma)* (cdf_nor(d)-pow(I/s,k)*cdf_
        nor(d-(2.*log(I/s))/(sigma*sqrt(t)))));

    return(res);
}

/*Price Formula*/
static double FormulaBjS(double r,double divid,double sigma
    ,double t,double s,double k)
{
    double alpha,beta,I,b,b1,b0,h,res,call_price,call_delta;

    b=r-divid;
    if (b>=r)
    {
        pnl_cf_call_bs(s,k,t,r,divid,sigma,&call_price,&call_
            delta);
        res=call_price;
    }
    else
    {
        b=r-divid;
        beta=(0.5-b/SQR(sigma))+sqrt(SQR((b/SQR(sigma)-0.5))+
            2.0*r/SQR(sigma));
        b1=k*beta/(beta-1.0);
    }
}

```

```

        b0=MAX(k,k*r/(r-b));
        h=-(b*t+2*sigma*sqrt(t))*(b0/(b1-b0));
        I=b0+(b1-b0)*(1-exp(h));
        alpha=(I-k)*pow(I,-beta);
        if (s>=I) res=s-k;
        else
res=alpha*pow(s,beta)-alpha*phi(r,divid,sigma,s,t,beta,
    I,I)+phi(r,divid,sigma,s,t,1,I,I)-
    phi(r,divid,sigma,s,t,1,k,I)-k*phi(r,divid,sigma,s,t,0
    ,I,I)+k*phi(r,divid,sigma,s,t,0,k,I);
    }

    return res;
}

/* Bjerksund-Stensland AP*/
static int BjerksundStensland_92(double s,NumFunc_1*p,
    double t,double r,double divid,double sigma,double *ptprice,
    double *ptdelta)
{
    double s_plus,s_minus;

    s_plus=s*(1.+INC);
    s_minus=s*(1.-INC);
    if ((p->Compute)==&Call)/*Call Case*/
    {
        /*Price*/
        *ptprice=FormulaBjS(r,divid,sigma,t,s,p->Par[0].Val.
        V_PDOUBLE);

        /*Delta*/
        *ptdelta=(FormulaBjS(r,divid,sigma,t,s_plus,p->Par[0]
        .Val.V_PDOUBLE)-(FormulaBjS(r,divid,sigma,t,s_minus,p->
        Par[0].Val.V_PDOUBLE)))/(2.*s*INC);
    }
    else
        if ((p->Compute)==&Put)/*Put Case*/
        {
            /*Price*/
            *ptprice=FormulaBjS(divid,r,sigma,t,p->Par[0].Val.V_PDO
            UBLE,s);

```

```

/*Delta*/
*ptdelta=(FormulaBjS(divid,r,sigma,t,p->Par[0].Val.V_PD0
    UBLE,s_plus)-FormulaBjS(divid,r,sigma,t,p->Par[0].Val.V_PD0
    UBLE,s_minus))/(2.*s*INC);
    }
    else
        return OPTION_IRRELEVANT_TO_THIS_METHOD;
return OK;
}

int CALC(AP_Bjerk SundStensland)(void *Opt,void *Mod,Pricing
    Method *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 Bjerk SundStensland_92(ptMod->S0.Val.V_PD0UBLE,
        ptOpt->PayOff.Val.V_NUMFUNC_1,ptOpt->Matu
        rity.Val.V_DATE-ptMod->T.Val.V_DATE,r,divid,
        ptMod->Sigma.Val.V_PD0UBLE,
        &(Met->Res[0].Val.V_DOUBLE),&(Met->Res[1].
        Val.V_DOUBLE));
}

static int CHK_OPT(AP_Bjerk SundStensland)(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_Bjerk SundStensland)=
{
    "AP_Bjerk SundStensland",
    {{ " ",PREMIA_NULLTYPE,{0},FORBID}},
    CALC(AP_Bjerk SundStensland),
    {{ "Price",DOUBLE,{100},FORBID},{ "Delta",DOUBLE,{100},FORB
        ID} ,{" ",PREMIA_NULLTYPE,{0},FORBID}},
    CHK_OPT(AP_Bjerk SundStensland),
    CHK_ok ,
    MET(Init)
} ;
```

References