

[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;
    int dummy;
    do {
        previous = current;
        dummy=Put_BlackScholes_73(previous,K,T,r,div
            id,sigma,&put_price,&put_delta);
        current=K-put_price;
    } while(!(fabs((previous-current)/current)<=pr
        ecision));
    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;
    int dummy;
    do {
        previous=current;
        dummy=Call_BlackScholes_73(previous,K,T,r,div
            id,sigma,&call_price,&call_delta);
        current=K+call_price;
    } while(!(fabs((previous-current)/current)<=pr
        ecision));
    return current;
}

/* 2-points Bunch-Johnsonn AP*/
```

```

static int BunchJohnsonn_92(double s,NumFunc_1*p,
    double t,double r,double divid,double sigma,double *pt
    price,double *ptdelta)

{
    int dummy;
    double p1,p2,crit12,k,price,delta,val,w1,w2,d1,
        d2,d1c,d2c;

    k=p->Par[0].Val.V_PDDOUBLE;
    if ((p->Compute)==&Call)
    {
        val=-1.;
        dummy=Call_BlackScholes_73(s,k,t,r,divid,sig
            ma,&price,&delta);
        p1=price;
        crit12=CallCriticalPrice(r,divid,sigma,t/2.,k
            );
    }
    else
    {
        val=1.;
        dummy=Put_BlackScholes_73(s,k,t,r,divid,sig
            ma,&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)/(sig
        ma*sqrt(t));
    d2=d1-sigma*sqrt(t);
    w1=exp(-divid*t/2.)*N(-val*d1c)+exp(-divid*t)*
        NN(val*d1c,-val*d1,-sqrt(0.5));
    w2=exp(-r*t/2.)*N(-val*d2c)+exp(-r*t)*NN(val*d2
        c,-val*d2,-sqrt(0.5));
    p2=val*(k*w2-s*w1);

    /*Price*/

```

```

    *ptprice=2.*p2-p1;

    /*Delta*/
    *ptdelta=-2.*val*w1-delta;

    return OK;
}

int CALC(AP_BunchJohnsonn)(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 BunchJohnsonn_92(ptMod->S0.Val.V_PDOUBLE,
        E,
        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) );
}

int CHK_OPT(AP_BunchJohnsonn)(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)

```

```
{
  return OK;
}

PricingMethod MET(AP_BunchJohnsonn)=
{
  "AP_BunchJohnsonn",
  {{ " ",END,0,FORBID}},
  CALC(AP_BunchJohnsonn),
  {{ "Price",DOUBLE,100,FORBID},{ "Delta",DOUBLE,
  100,FORBID} ,{ " ",END,0,FORBID}},
  CHK_OPT(AP_BunchJohnsonn),
  CHK_ok ,
  MET(Init)
} ;
```

References