

[Help](#)

```
#include "bs1d_std.h"

/*Put McMillan Exponent*/
static double McMillanPut_Exp(double r,double divid,
double sigma,double T)
{
double ratio = 2.0 * (r-divid) / (sigma * sigma);
double delta = (ratio - 1.0);

delta=SQR(delta)+8.0*(1.0+r*T)/(sigma*sigma*T);

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

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

delta=SQR(delta)+8.0*(1.0+r*T)/(sigma*sigma*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;
int dummy;

do {
    previous = current;
    dummy=Put_BlackScholes_73(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))-exp
onent);
    } while(!(fabs((previous-current)/current)
t)<=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;
int dummy;

do {
    previous =current;
    dummy=Call_BlackScholes_73(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))+exp
onent);
    } while(!(fabs((previous-current)/current)
t)<=precision));

```

```

    return current;
}

/*McMillan Formula*/
static double Formula_McMillan(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;
    int dummy;

    if ((p->Compute)==&Put)
    {
        critical_price=Contact_PointPut(r,divid,
sigma,T,K,McMillanPut_Exp);
        if(x < critical_price)
        {
            return (K - x);
        }
        else
        {
            exponent=McMillanPut_Exp(r,divid,sig
ma,T);
            a=critical_price*(1.-exp(-divid*T)*Nd
1(critical_price,r,divid,-sigma,T,K))/(-exponen
t);
            dummy=Put_BlackScholes_73(x,K,T,r,div
id,sigma,&put_price,&put_delta);
            return put_price+a*pow(x/critical_pr
ice,exponent);
        }
    }
    else
        if ((p->Compute)==&Call)
        {
            critical_price=Contact_PointCall(r,
divid,sigma,T,K,McMillanCall_Exp);

```

```

        if(x >= critical_price)
        {
            return (x - K);
        }
        else
        {
            exponent=McMillanCall_Exp(r,divid
,sigma,T);
            a=critical_price*(1.-exp(-divid*
T)*Nd1(critical_price,r,divid,sigma,T,K))/exponen
t;
            dummy=Call_BlackScholes_73(x,K,T,
r,divid,sigma,&call_price,&call_delta);
            return call_price+a*pow(x/critic
al_price,exponent);
        }
    }

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

static int McMillan_81(double s,NumFunc_1 *p,
double t,double r,double divid,double sigma,double *pt
price,double *ptdelta){
double s_plus,s_minus;

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

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

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

return OK;

```

```

    }

int CALC(AP_McMillan)(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 McMillan_81(ptMod->S0.Val.V_PDOUBLE,
        ptOpt->PayOff.Val.V_NUMFUNC_1,ptOpt->Matu
        rity.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_McMillan)(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_McMillan)=
{
    "AP_McMillan",
    {{" ",END,0,FORBID}},
    CALC(AP_McMillan),

```

```
{{"Price",DOUBLE,100,FORBID},{"Delta",DOUBLE,
100,FORBID} ,{" ",END,0,FORBID}},
CHK_OPT(AP\_McMillan),
CHK_ok ,
MET(Init)
} ;
```

References