

[Help](#)

```
#include "bs1d_lim.h"

static int CallUpOut_ReinerRubinstein(double s,
    double k,double l,double rebate,double t,double r,
    double divid,double sigma,double *ptprice,double *ptde
    lta)
{
    int phi,eta;
    double A,B,C,D,E,F;
    double dA,dB,dC,dD,dE,dF;

    phi=1;
    eta=-1;
    formula(s,k,r,divid,sigma,t,l,rebate,phi,eta,
    &A,&B,&C,&D,&E,&F,
        &dA,&dB,&dC,&dD,&dE,&dF);
    if (k>=1)
    {
        *ptprice=F;
        *ptdelta=dF;
    }
    else
    {
        *ptprice=A-B+C-D+F;
        *ptdelta=dA-dB+dC-dD+dF;
    }
    return OK;
}

int CALC(CF_CallUpOut)(void*Opt,void *Mod,Pricing
    Method *Met)
{
    TYPEOPT* ptOpt=( TYPEOPT*)Opt;
    TYPEMOD* ptMod=( TYPEMOD*)Mod;
    double r,divid,limit,rebate;

    r=log(1.+ptMod->R.Val.V_DOUBLE/100.);
    divid=log(1.+ptMod->Divid.Val.V_DOUBLE/100.);
    limit=((ptOpt->Limit.Val.V_NUMFUNC_1)->Compu
    te)((ptOpt->Limit.Val.V_NUMFUNC_1)->Par,ptMod->T.
```

```

    Val.V_DATE);
    rebate=((ptOpt->Rebate.Val.V_NUMFUNC_1)->
    Compute)((ptOpt->Rebate.Val.V_NUMFUNC_1)->Par,ptMod->T.Val.V_DATE);

    return CallUpOut_ReinerRubinstein(ptMod->S0.
    Val.V_PDOUBLE,(ptOpt->PayOff.Val.V_NUMFUNC_1)->
    Par[0].Val.V_PDOUBLE,
        limit,rebate,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(CF_CallUpOut)(void *Opt, void *Mod)
{Option* ptOpt=(Option*)Opt;
    TYPEOPT* opt=(TYPEOPT*)(ptOpt->TypeOpt);

    if ((opt->Parisian).Val.V_BOOL==WRONG)
    return strcmp( ((Option*)Opt)->Name,"
    CallUpOutEuro");
    return WRONG;

}

static int MET(Init)(PricingMethod *Met)
{
    return OK;
}

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

```

} ;

References