

[Help](#)

```
#include "mer1d_std.h"

static int Call_Merton(double x, NumFunc_1 *p, double T,
    double r, double divid, double sigma, double lambda, double m,
    double v, double *ptprice, double *ptdelta)
{

    double lambdaT, mv2, exmv2, EU, mu, M, sigma02, sigmasqrt, price,
        delta, test, puissancen1, factorieln, n, d1, d2, sigma2, rT, muT, ex_
        r_lT, K;

    lambdaT=lambda*T;
    K=p->Par[0].Val.V_DOUBLE;
    mv2=m+v/2.;
    exmv2=exp(mv2);
    EU=exmv2-1;
    mu=r-divid-lambda*EU;
    rT=r*T;
    muT=mu*T;
    M=exp(T*(-divid-lambda*exmv2));
    sigma02=sigma*sigma;
    sigmasqrt=sigma*sqrt(T);
    d1=(log(x/K)+sigma02*T/2+muT)/sigmasqrt;
    d2=d1-sigmasqrt;
    price=x*M*cdf_nor(d1)-K*exp(-rT-lambdaT)*cdf_nor(d2);
    puissancen1=1.;
    factorieln=1.;
    delta=M*cdf_nor(d1);
    ex_r_lT=exp(-rT-lambdaT);
    test=exp(-lambdaT);
    puissancen1=1.;
    factorieln=1.;
    n=0;

    while (test<0.99999)
    {n++;

        factorieln*=n;/*  n!  */
        puissancen1*=lambdaT;/* (lambda*T)^n */
    }
}
```

```

        sigma2=sigma02+v*(double)n/T;
        sigmasqrt=sqrt(sigma2*T);
        d1=(log(x/K)+sigma2*T/2+n*(mv2)+muT)/sigmasqrt;
        d2=d1-sigmasqrt;
        price+=(puissancen1/factorieln)*(x*exp(n*mv2)*M*cdf_
nor(d1)-K*ex_r_lT*cdf_nor(d2));
        test+=exp(-lambdaT)*puissancen1/factorieln;

        delta+=(puissancen1/factorieln)*exp(n*mv2)*M*cdf_nor(
d1);
    }

    *ptprice=price;
    *ptdelta=delta;

    return OK;
}

int CALC(CF_Call_Merton)(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 Call_Merton(ptMod->S0.Val.V_PDOUBLE,ptOpt->PayO
ff.Val.V_NUMFUNC_1,ptOpt->Maturity.Val.V_DATE-ptMod->T.Val.
V_DATE,
        r,divid,ptMod->Sigma.Val.V_PDOUBLE,ptMod->Lambd
a.Val.V_PDOUBLE,ptMod->Mean.Val.V_PDOUBLE,ptMod->Variance.
Val.V_PDOUBLE,&(Met->Res[0].Val.V_DOUBLE),&(Met->Res[1].Val.
V_DOUBLE));
}

static int CHK_OPT(CF_Call_Merton)(void *Opt, void *Mod)
{
    /*

```

```

    Option* ptOpt=(Option*)Opt;
    TYPEOPT* opt=(TYPEOPT*)(ptOpt->TypeOpt);
    */
    return strcmp( ((Option*)Opt)->Name,"CallEuro");
}

static int MET(Init)(PricingMethod *Met,Option *Opt)
{
    if ( Met->init == 0)
    {
        Met->init=1;
    }

    return OK;
}

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

```

References