

Help

```

#include <stdlib.h>
#include "svj.h"
#include "merhes1d_std.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
    (2007+2) //The "#else" part of the code will be freely av
    ailable after the (year of creation of this file + 2)
static int CHK_OPT(CF_PutMertonHeston)(void *Opt, void *
    Mod)
{
    return NONACTIVE;
}
int CALC(CF_PutMertonHeston)(void *Opt, void *Mod, Pricing
    Method *Met)
{
    return AVAILABLE_IN_FULL_PREMIA;
}
#else
int CFPutMertonHeston(double St0, NumFunc_1 *p, double T,
    double r, double divid, double V0,double kappa,double theta,
    double sigmav,double rho,double lambda, double m0,double v,
    double *ptprice, double *ptdelta)
{
    double K,price,delta;
    double stdv;
    SVJPARAMS *svj;

    stdv=sqrt(v);
    K=p->Par[0].Val.V_DOUBLE;
    svj = malloc(sizeof(SVJPARAMS));
    svj->heston = 1;
    svj->merton = 1;
    svj->phi = -1.;
    svj->type_f = 1;
    svj->K = K;
    svj->St0 = St0;
    svj->T = T;
    svj->r = r;
    svj->divid = divid;

```

```

    svj->sigmav = sigmav;
    svj->V0      = V0;
    svj->theta   = theta;
    svj->rho     = rho;
    svj->kappa   = kappa;
    svj->lambda  = lambda;
    svj->m0      = m0;
    svj->v       = stdv;
    calc_price_svj(svj,&price,&delta);

    /* Price */
    *ptprice=price;

    /* Delta */
    *ptdelta=delta;

    free(svj);
    return OK;
}

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

    if(ptMod->Sigma.Val.V_PDDOUBLE==0.0)
    {
        Fprintf(TOSCREEN,"BLACK-SHOLES MODEL{n{n{n");
        return WRONG;
    }
    else
    {
        r=log(1.+ptMod->R.Val.V_DOUBLE/100.);
        divid=log(1.+ptMod->Divid.Val.V_DOUBLE/100.);

        return CFPutMertonHeston(ptMod->S0.Val.V_PDDOUBLE,
            ptOpt->PayOff.Val.V_NUMFUNC_1,
            ptOpt->Maturity.Val.V_DATE-ptMod->T.Val.V_
DATE,

```

```

        r,
        divid, ptMod->Sigma0.Val.V_PDOUBLE
        ,ptMod->MeanReversion.hal.V_PDOUBLE,
        ptMod->LongRunVariance.Val.V_PDOUBLE,
        ptMod->Sigma.Val.V_PDOUBLE,
        ptMod->Rho.Val.V_PDOUBLE,
        ptMod->Lambda.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_PutMertonHeston)(void *Opt, void *
    Mod)
{
    return strcmp( ((Option*)Opt)->Name,"PutEuro");
}

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

    return OK;
}
PricingMethod MET(CF_PutMertonHeston)=
{
    "CF_Put_MerHes",
    {{" ",PREMIA_NULLTYPE,{0},FORBID}},
    CALC(CF_PutMertonHeston),
    {"Price",DOUBLE,{100},FORBID},
    {"Delta",DOUBLE,{100},FORBID} ,
    {" ",PREMIA_NULLTYPE,{0},FORBID}},
    CHK_OPT(CF_PutMertonHeston),
    CHK_ok,

```

```
    MET(Init)  
};
```

References