

Help

```
extern "C"{
#include "mer1d_lim.h"
#include "enums.h"
}
#include "math/levy_fd.h"
extern "C"{

#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(FD_ImpExpDownOut)(void *Opt, void *Mod)
{
    return NONACTIVE;
}
int CALC(FD_ImpExpDownOut)(void *Opt,void *Mod,Pricing
    Method *Met)
{
    return AVAILABLE_IN_FULL_PREMIA;
}
#else
static int ImpExpDownOut(int am,double S0,NumFunc_1 *p,
    double l_down,double rebate,double T,double r,double divid,
    double sigma,double lambda,double mu,double gamma2,double dx,
    int M,int flag_scheme,double *ptprice,double *ptdelta)
{

    double price0,delta0;
    int flag_callput,flag_stdbarrier;

    /*Construction of the model*/
    double delta=sqrt(gamma2);
    double ldownlog=log(l_down/S0);

    if(dx>fabs(ldownlog)/2.)
        dx=fabs(ldownlog)/2.;

    int N1 = (int)ceil(fabs(ldownlog)/dx);
    dx = fabs(ldownlog)/N1;

    double A1 = ldownlog+dx;
```

```

Merton_measure measure(mu,delta,lambda,sigma,dx);
double k = 3;
double Ar = log(2.) + r*T + k*sqrt(T*measure.varX1);

if (Ar>30) Ar = 30;
int N = (int) ceil((Ar-lndownlog)/dx);
Ar = lndownlog + N*dx;
double K=p->Par[0].Val.V_DOUBLE;

flag_stdbarrier=3;

/*Price Computation*/
if ((p->Compute)==&Put)
{
    flag_callput=2;

    if (flag_scheme==1)
        vector<double> u = price2(am,measure,flag_callput,
flag_stdbarrier,r,divid,S0,K,rebate,A1,Ar,N,T,M,price0,delta0);
    else
        vector<double> u = price2c(am,measure,flag_callput,
flag_stdbarrier,r,divid,S0,K,rebate,A1,Ar,N,T,M,price0,delta0);

    /*Price */
    *ptprice=price0;

    /*Delta */
    *ptdelta=delta0;
}
else
if ((p->Compute)==&Call)
{
    /*Price */
    flag_callput=1;
    if (flag_scheme==1)
        vector<double> u = price2(am,measure,flag_callput
,flag_stdbarrier,r,divid,S0,K,rebate,A1,Ar,N,T,M,price0,delta0);
    else

```

```

        vector<double> u = price2c(am,measure,flag_callp
ut,flag_stdbarrier,r,divid,S0,K,rebate,Al,Ar,N,T,M,price0,
delta0);

        *ptprice=price0;

        /*Delta */
        *ptdelta=delta0;
    }

    return OK;
}

int CALC(FD_ImpExpDownOut)(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)->Compute)((ptOpt->Limit.Val.V_NUMFUN
    rebate=((ptOpt->Rebate.Val.V_NUMFUNC_1)->Compute)((ptOpt->Rebate.Val.V_NUMFUNC_1)->Par,ptMod->T.Val.V_DATE);

    return ImpExpDownOut(ptOpt->EuOrAm.Val.V_BOOL,ptMod->S0.
        Val.V_PDOUBLE,
                        ptOpt->PayOff.Val.V_NUMFUNC_1, limi
        t,rebate,ptOpt->Maturity.Val.V_DATE-ptMod->T.Val.V_DATE,r,
        divid,ptMod->Sigma.Val.V_PDOUBLE,ptMod->Lambda.Val.V_PDOUB
        LE,ptMod->Mean.Val.V_PDOUBLE,ptMod->Variance.Val.V_PDOUBLE,
        Met->Par[0].Val.V_DOUBLE,Met->Par[1].Val.V_INT,Met->Par[2].
        Val.V_ENUM.value,&(Met->Res[0].Val.V_DOUBLE),&(Met->Res[1].
        Val.V_DOUBLE));
}

static int CHK_OPT(FD_ImpExpDownOut)(void *Opt, void *Mod)
{
    Option* ptOpt=(Option*)Opt;
    TYPEOPT* opt=(TYPEOPT*)(ptOpt->TypeOpt);

```

```

    if ((opt->OutOrIn).Val.V_BOOL==OUT)
        if ((opt->DownOrUp).Val.V_BOOL==DOWN)
            if ((opt->EuOrAm).Val.V_BOOL==EURO)
                if ((opt->Parisian).Val.V_BOOL==WRONG)
                    return OK;

    return WRONG;
}
#endif //PremiaCurrentVersion

static int MET(Init)(PricingMethod *Met,Option *Opt)
{
    static int first=1;

    if (first)
    {
        Met->Par[0].Val.V_PDOUBLE=0.001;
        Met->Par[1].Val.V_INT2=100;
        Met->Par[2].Val.V_ENUM.value=1;
        Met->Par[2].Val.V_ENUM.members=&PremiaEnumExpPart;
        first=0;
    }

    return OK;
}

PricingMethod MET(FD_ImpExpDownOut)=
{
    "FD_ImpExpDownOut",
    {{"Space Discretization Step",DOUBLE,{500},ALLOW},{ "TimeS
        tepNumber",INT2,{100},ALLOW},
        {"Explicit Part",ENUM,{100},ALLOW},
        {" ",PREMIA_NULLTYPE,{0},FORBID}}},
    CALC(FD_ImpExpDownOut),
    {{"Price",DOUBLE,{100},FORBID},{ "Delta",DOUBLE,{100},FORB
        ID},{ " ",PREMIA_NULLTYPE,{0},FORBID}}},
    CHK_OPT(FD_ImpExpDownOut),
    CHK_split,
    MET(Init)
}

```

};

}

References