

```

    Help
extern "C"{
#include "nig1d_std.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_ImpExp)(void *Opt, void *Mod)
{
    return NONACTIVE;
}
int CALC(FD_ImpExp)(void *Opt,void *Mod,PricingMethod *Met)
{
    return AVAILABLE_IN_FULL_PREMIA;
}
#else

static int ImpExp(int am,double S0,NumFunc_1 *p,double T,
    double r,double divid,double sigma,double theta,double kappa,
    double dx,int M,int flag_scheme,double *ptprice,double *ptdelta)
{

    double price0,delta0;
    int flag_callput,flag_stdbarrier;
    double rebate=0.;

    /*Construction of the model*/
    NIG_measure measure(theta, sigma, kappa, dx);

    double K=p->Par[0].Val.V_DOUBLE;

    /*Localization on a bounded computational domain (-A1,Ar)
    */
    double k = 3;
    double A1 = log(2./3) + T*measure.espX1 - k*sqrt(T*measu
        re.varX1);
    double Ar = log(2.) + r*T + k*sqrt(T*measure.varX1);

```

```

    if (A1<-30) A1 = -30;
    if (Ar>30) Ar = 30;
    int Nl = (int)ceil(-A1/dx);
    int Nr = (int)ceil(Ar/dx);
    int N = Nl+Nr;
    A1 = -Nl*dx;
    Ar = Nr*dx;

    if ((p->Compute)==&Put)
        flag_callput=2;
    else /*if ((p->Compute)==&Call)*/
        flag_callput=1;

    flag_stdbarrier=1;

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

    /*Price */
    *ptprice=price0;

    /*Delta */
    *ptdelta=delta0;

    return OK;
}

int CALC(FD_ImpExp)(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 ImpExp(ptOpt->EuOrAm.Val.V_BOOL,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_SPDOUBLE,ptMod->Theta.Val.V_DOUBLE,ptMod->Kappa.Val.V_
SPDOUBLE,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_ImpExp)(void *Opt, void *Mod)
{

    if ( (strcmp( ((Option*)Opt)->Name,"CallEuro")==0) || (
        strcmp( ((Option*)Opt)->Name,"PutEuro")==0) || (strcmp( ((
        Option*)Opt)->Name,"CallAmer")==0) || (strcmp( ((Option*)Opt)->
        Name,"PutAmer")==0))

        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_ImpExp)=
{
    "FD_ImpExp",
    {"Space Discretization Step",DOUBLE,{500},ALLOW},{"TimeStepNumber",INT2,{100},ALLOW},
    {"Explicit Part",ENUM,{100},ALLOW},
    {" ",PREMIA_NULLTYPE,{0},FORBID}},
    CALC(FD_ImpExp),
    {"Price",DOUBLE,{100},FORBID},{"Delta",DOUBLE,{100},FORBID},{"ID"},{" ",PREMIA_NULLTYPE,{0},FORBID}},
    CHK_OPT(FD_ImpExp),
    CHK_split,
    MET(Init)
};
}
```

References