```
   Help
extern "C"{
#include  "temperedstable1d_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_ImpExpUpOut)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(FD_ImpExpUpOut)(void *Opt,void *Mod,PricingMethod
    *Met)
{
return AVAILABLE_IN_FULL_PREMIA;
}
#else

static int ImpExpUpOut(int am,double S0,NumFunc_1  *p,
    double l_up,double rebate,double T,double r,double divid,double
    alphap,double alpham,double lambdap,double lambdam,double
    cp,double cm,double dx,int M,int flag_scheme,double *pt
    price,double *ptdelta)
{

  double price0,delta0;
  int flag_callput,flag_stdbarrier;

   double luplog=log(l_up/S0);
   if(dx>fabs(luplog)/2.)
    dx=fabs(luplog)/2.;
  int Nl = (int)ceil(fabs(luplog)/dx);
  dx = fabs(luplog)/Nl;

  double Ar = luplog-dx;

   /*Construction of the model*/
  TS_measure measure(alphap,alpham,lambdap,lambdam,cp,cm,dx
```

```
   );

double k = 3;
 double Al = log(2./3) + T*measure.espX1 - k*sqrt(T*measu
   re.varX1);

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

 flag_stdbarrier=2;


 /*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,
 else
  vector<double> u = price2c(am,measure,flag_callput,flag_     stdbarrier,r,divi
     /*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,
 else
  vector<double> u = price2c(am,measure,flag_callput,flag_     stdbarrier,r,divi
 *ptprice=price0;

 /*Delta */
 *ptdelta=delta0;
```

```
      }


   return OK;
}

int CALC(FD_ImpExpUpOut)(void *Opt,void *Mod,PricingMethod
   *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)((pt
    Opt->Rebate.Val.V_NUMFUNC_1)->Par,ptMod->T.Val.V_DATE);

  return ImpExpUpOut(ptOpt->EuOrAm.Val.V_BOOL,ptMod->S0.Val
    .V_PDOUBLE,
    ptOpt->PayOff.Val.V_NUMFUNC_1, limit,rebate,ptOpt->
    Maturity.Val.V_DATE-ptMod->T.Val.V_DATE,r,divid,ptMod->Alpha
    Plus.Val.V_PDOUBLE,ptMod->AlphaMinus.Val.V_PDOUBLE,ptMod->
    LambdaPlus.Val.V_PDOUBLE,ptMod->LambdaMinus.Val.V_PDOUBLE,pt
    Mod->CPlus.Val.V_PDOUBLE,ptMod->CMinus.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_ImpExpUpOut)(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==UP)
        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_ImpExpUpOut)=
{
  "FD_ImpExpUpOut",
  {{"Space Discretization Step",DOUBLE,{500},ALLOW},
   {"TimeStepNumber",INT2,{100},ALLOW},
   {"Explicit Part",ENUM,{100},ALLOW},
   {" ",PREMIA_NULLTYPE,{0},FORBID}},
  CALC(FD_ImpExpUpOut),
  {{"Price",DOUBLE,{100},FORBID},{"Delta",DOUBLE,{100},FORB
    ID},{" ",PREMIA_NULLTYPE,{0},FORBID}},
  CHK_OPT(FD_ImpExpUpOut),
  CHK_split,
  MET(Init)
};

}
```

References