```
   Help
extern "C"{
#include "mer1d_std.h"
}
#include "math/levy.h"
#include "math/fft.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(AP_Carr)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(AP_Carr)(void*Opt,void *Mod,PricingMethod *Met)
{
return AVAILABLE_IN_FULL_PREMIA;
}
#else
  static int CarrMadanMer(double S0,NumFunc_1  *p,double T,
    double r,double divid,double sigma,double lambda,double mu,
    double gamma2,double *ptprice,double *ptdelta)
  {
    double K;
    double Sigma=0.2;
    double delta=sqrt(gamma2);

    /*Construction of the model*/
    Merton_measure measure(mu,delta,lambda,sigma,0.1);
    K=p->Par[0].Val.V_DOUBLE;

    S0 *= exp(-divid*T); //taking account of dividends
    int Nlimit =  2048;  //number of integral discretizati
    on steps
    double logstrikestep = 0.01;
    double k0 = log(K/S0);
    double h = 2*M_PI/Nlimit/logstrikestep; //integral dis
    cretization step
    double A = (Nlimit-1)*h; // integration domain is (-A/2
    ,A/2)
```

```
double* z = new double [Nlimit];
double* z_img = new double [Nlimit];
double* y = new double [Nlimit];
double* y_img = new double [Nlimit];

double vn = -A/2;
//double weight = 0.5; //trapezoidal rule weights
double weight = 1./3; //Simpson's rule weights

//delta
complex<double> dzeta = exp(I*vn*(r*T-k0))*(measure.cf(
T,vn-I)
                                                -exp(-T*Si
gma*Sigma/2*(vn-I)*vn))/(I*vn);
z[0] = weight*real(dzeta);
z_img[0] = weight*imag(dzeta);

//price
y[0] = weight*real(dzeta/(1.+I*vn));
y_img[0] = weight*imag(dzeta/(1.+I*vn));
for(int n=1; n<Nlimit-1; n++){
  vn += h;
  //weight = 1; //trapezoidal rule weights
  weight = (weight<1) ? 4./3 : 2./3; //Simpson's rule
weights

  //delta
  dzeta = exp(I*vn*(r*T-k0))*(measure.cf(T,vn-I)
                                -exp(-T*Sigma*Sigma/2*(vn
-I)*vn))/(I*vn);
  z[n] = weight*real(dzeta);
  z_img[n] = weight*imag(dzeta);

  //price
  y[n] = weight*real(dzeta/(1.+I*vn));
  y_img[n] = weight*imag(dzeta/(1.+I*vn));
}
vn += h;
//weight = 0.5; //trapezoidal rule weights
weight = 2./3; //Simpson's rule weights
```

```
//delta
dzeta = exp(I*vn*(r*T-k0))*(measure.cf(T,vn-I)
                            -exp(-T*Sigma*Sigma/2*(vn-
I)*vn))/(I*vn);
z[Nlimit-1] = weight*real(dzeta);
z_img[Nlimit-1] = weight*imag(dzeta);

//price
y[Nlimit-1] = weight*real(dzeta/(1.+I*vn));
y_img[Nlimit-1] = weight*imag(dzeta/(1.+I*vn));

fft1d(z,z_img,Nlimit,-1);
fft1d(y,y_img,Nlimit,-1);

//Black-Scholes formula
double d1 = (log(S0/K) + (r+Sigma*Sigma/2)*T)/Sigma/sq
rt(T);
double d2 = d1 - Sigma*sqrt(T);
double CallBS = S0*normCDF(d1) - K*exp(-r*T)*normCDF(d2
);
double DeltaBS = normCDF(d1);

/*Call Case*/
*ptprice = CallBS + S0*A/2/M_PI/(Nlimit-1)*y[0];
*ptdelta = exp(-divid*T)*(DeltaBS + A/2/M_PI/(Nlimit-1)
*z[0]);

/*Put Case via parity*/
if ((p->Compute)==&Put)
  {
    *ptprice =*ptprice-S0+K*exp(-r*T);
    *ptdelta =*ptdelta-exp(-divid*T);
  }
//memory desallocation
delete [] z;
delete [] z_img;
delete [] y;
delete [] y_img;

return OK;
```

```
  }

  int CALC(AP_Carr)(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  CarrMadanMer(ptMod->S0.Val.V_PDOUBLE,ptOpt->
    PayOff.Val.V_NUMFUNC_1,ptOpt->Maturity.Val.V_DATE-ptMod->T.
    Val.V_DATE,r,divid,ptMod->Sigma.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(AP_Carr)(void *Opt, void *Mod)
  {
  if ( (strcmp( ((Option*)Opt)->Name,"CallEuro")==0) || (
    strcmp( ((Option*)Opt)->Name,"PutEuro")==0) )
    return OK;

  return WRONG;
  }


#endif //PremiaCurrentVersion
  static int MET(Init)(PricingMethod *Met,Option *Opt)
  {
    return OK;
  }

  PricingMethod MET(AP_Carr)=
  {
    "AP_Carr_Mer",
    {{" ",PREMIA_NULLTYPE,{0},FORBID}},
    CALC(AP_Carr),
    {{"Price",DOUBLE,{100},FORBID},{"Delta",DOUBLE,{100},FO
```

```
    RBID},{" ",PREMIA_NULLTYPE,{0},FORBID}},
    CHK_OPT(AP_Carr),
    CHK_ok,
    MET(Init)
  } ;
}
```

References