```
   Help
extern "C"{
#include "temperedstable1d_vol.h"
}
#include "math/levy.h"
#include "math/numerics.h"
#include "math/fft.h"
#include "math/intg.h"

extern "C"{

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
     (2008+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_CGMY_REALVAR)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(AP_CGMY_REALVAR)(void *Opt,void *Mod,Pricing
    Method *Met)
{
return AVAILABLE_IN_FULL_PREMIA;
}
#else

  static double ap, am;
  static double lap, lam;
  static double cpp, cmm;
  static double gamma2p, gamma2m;
  static double par[5];

  static double intfunim(double x);
  static double intfunre(double x);
  static complex<double> iint(double la, double v, double
    s, double al);
  static complex<double> logint(double la, double v,
    double s, double al, double gam);
  static complex<double> cphi(double v, double s, double T)
    ;
```

```
/*////////////////////////////////////////////////////*/
int ap_cgmy_realvar(int ifCall, double S0, double Strike,
   double T, double r, double divid, double alphap, double
  alpham,double lambdap,double lambdam,double cp,double cm,
  double parsigma, double parstep, int exp2, double *ptprice)
{

  double K;
  double sigma=parsigma;
  double temp;
  long int n;
 complex<double> fact;

  ap=alphap;
  am=alpham;
  lap=lambdap;
  lam=lambdam;
  cpp=cp;
  cmm=cm;

  K=Strike;//p->Par[0].Val.V_DOUBLE;
  K=K*K*T/10000.0;

  gamma2p=tgamma(2.0-ap);
  gamma2m=tgamma(2.0-am);
  double lpnu=exp((2.0-ap)*log(lap));
  double lmnu=exp((2.0-am)*log(lam));

  long int Nlimit;
  for(n=1,Nlimit=1;n<exp2+1;n++, Nlimit*=2);  //number of
   integral discretization steps
  double h=parstep;//step of integrtion

  double logstrikestep= 2*M_PI/Nlimit/h; //strike discret
  ization step
  double A = Nlimit*h/2.0; // integration domain is (-A/2
  ,A/2)
  double odd=-1.0;
  double mval=T*(cpp*gamma2p/lpnu+cmm*gamma2m/lmnu);

  double* y = new double [Nlimit];
```

```
double* y_img = new double [Nlimit];
   double* k_arr=new double[Nlimit];


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

complex<double> dzeta = exp(-r*T)*(cphi(vn, sigma, T))/
 ((sigma+I*vn)*(sigma+I*vn));///2.0;

y[0] = weight*real(dzeta);
y_img[0] = weight*imag(dzeta);
k_arr[0] = K;


//price
for(n=1; n<Nlimit-1; n++){
  vn += h;
  //weight = 1; //trapezoidal rule weights
  odd*= -1.0; //weight = (weight<1) ? 4./3 : 2./3; //Si
mpson's rule weights
  temp=h*n*K;

  dzeta = exp(-r*T)*exp(I*temp)*(cphi(vn, sigma, T) ) /
 ((sigma+I*vn)*(sigma+I*vn));
  //price
  y[n] = (1.0+odd*weight)*real(dzeta);
  y_img[n] = (1.0+odd*weight)*imag(dzeta);
  k_arr[n]=K+n*logstrikestep;
}

vn += h;
//weight = 0.5; //trapezoidal rule weights
weight = 1.0/3.0;//Simpson's rule weights

temp=h*n*K;
dzeta = exp(-r*T)*exp(I*temp)*(cphi(vn, sigma, T) ) / (
(sigma+I*vn)*(sigma+I*vn));

y[Nlimit-1] = weight*real(dzeta);
```

```
    y_img[Nlimit-1] = weight*imag(dzeta);
    k_arr[Nlimit-1] = K+(Nlimit-1)*logstrikestep;

    fft1d(y,y_img,Nlimit,1);
//*/

    if (ifCall)//((p->Compute)==&Call)
    {
     for(n=0;n<Nlimit-1;n++)
    {
      fact=exp((sigma-I*A)*k_arr[n])*A/M_PI;
  temp=y[n];
  y[n]=real(fact)*y[n]-imag(fact)*y_img[n] + exp(-r*T)*(mv
    al-k_arr[n]);
  y_img[n]=real(fact)*y_img[n]+imag(fact)*temp;
      y[n]=y[n]>0?sqrt(y[n]/T)*100.0:-1;
  k_arr[n]=sqrt(k_arr[n]/T)*100.0;
    }
    }
    else
    {
    for(n=0;n<Nlimit-1;n++)
    {
      fact=exp((sigma-I*A)*k_arr[n])*A/M_PI;
  temp=y[n];
  y[n]=real(fact)*y[n]-imag(fact)*y_img[n];
  y_img[n]=real(fact)*y_img[n]+imag(fact)*temp;
      y[n]=y[n]>0?sqrt(y[n]/T)*100.0:-1;
  k_arr[n]=sqrt(k_arr[n]/T)*100.0;
    }
    }

    *ptprice = y[0];//sqrt(res/T);

    delete [] y;
    delete [] y_img;
    delete [] k_arr;

    return OK;
  }
```

```
//-------------------------------------------------------
    --------
//-------------------------------------------------------
    ----
static complex<double> cphi(double v, double s, double T)
{
  return exp( -T*( cpp*logint(lap, v, s, ap, gamma2p) + cm
    m*logint(lam, v, s, am, gamma2m) ) );
}

/*-------------------------------------------------------
    ---*/
static complex<double> logint(double la, double v, double
    s, double al, double gam)
{
  complex<double> vs(2.0*s, 2.0*v);
  double fact1=1.0/al;
  double fact2=1.0/(1.0-al)/al;
  double fact3=1.0/(2.0-al);
  if(al<1.0)
  {
    return (vs*fact1+la*la*fact2)*iint(la, v, s, 1.0-al)
    + la*fact2*vs*iint(la, v, s, 2.0-al) - exp(al*log(la))/*tg
    amma(2.0-al)*/*gam*fact2;
  }
  else
  {
    return vs*fact1*fact3*( vs+la*la/(1.0-al) )*iint(la,
    v, s, 3.0-al) + la*fact2*fact3*( vs*(3.0-2.0*al)+la*la )*ii
    nt(la, v, s, 2.0-al) - exp(al*log(la))/*tgamma(2.0-al)*/*
    gam*fact2;
  }
}


//-------------------------------------------------------
    ----
static complex<double> iint(double la, double v, double s,
    double al)
{
        double re_res, im_res;
```

```
double re, im, err, lim, lim2;

lim=50.0;

if(fabs(v)>0.8*s)
{
par[0]=la;
par[1]=la;
par[2]=v>0?2.0*v:-2.0*v;//s;
par[3]=al;
par[4]=2.0*s;

lim=fabs(la/v);
lim2=sqrt(-log(2.0*fabs(v)*1.0e-15)/2.0/fabs(v));
if(lim<lim2) {lim=lim2;}
intg(0, lim, intfunre, 1e-14, 1e-10, &re, &err);
intg(0, lim, intfunim, 1e-14, 1e-10, &im, &err);

if(v<0.0) {im*=-1.0;}

double factr=exp((al+1.0)/2.0*log(2.0));
double alcos=cos((al+1.0)*M_PI/4.0);
double alsin=sin((al+1.0)*M_PI/4.0);
if(v>0.0) {alsin*=-1.0;}

re_res=factr*(alcos*re-alsin*im);
im_res=factr*(alcos*im+alsin*re);
}
else
{
  par[0]=la;
  par[1]=0.0;
  par[2]=s;
  par[3]=al;
  par[4]=v;

  intg(0, lim, intfunre, 1e-14, 1e-10, &re_res, &err);
  intg(0, lim, intfunim, 1e-14, 1e-10, &im_res, &err);
  im_res=-im_res;
}
complex<double> result( re_res, im_res );
```

```
  return result;
}

//*/-------------------------------------------------------
    --------
static double intfunre(double x)
{
  if(x==0)
      {return 0.0;}
  else
      {return exp(par[3]*log(x)-par[2]*x*x-par[0]*x)*cos(
    par[1]*x+par[4]*x*x);}
}

//-------------------------------------------------------
    ------
static double intfunim(double x)
{
  if((x==0)||((par[4]*par[4]+par[1]*par[1])==0))
      {return 0.0;}
  else
      {return exp(par[3]*log(x)-par[2]*x*x-par[0]*x)*sin(
    par[1]*x+par[4]*x*x);}
}

//-------------------------------------------------------
    ----------

  int CALC(AP_CGMY_REALVAR)(void *Opt,void *Mod,Pricing
    Method *Met)
  {
    TYPEOPT* ptOpt=(TYPEOPT*)Opt;
    TYPEMOD* ptMod=(TYPEMOD*)Mod;
    double r, divid, strike;
    NumFunc_1 *p;
    int ifCall;

    r=log(1.+ptMod->R.Val.V_DOUBLE/100.);
    divid=log(1.+ptMod->Divid.Val.V_DOUBLE/100.);
    p=ptOpt->PayOff.Val.V_NUMFUNC_1;
```

```
    ifCall=((p->Compute)==&Call);

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

    return ap_cgmy_realvar(
      ifCall, ptMod->S0.Val.V_PDOUBLE, strike, ptOpt->Matu
    rity.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_PDOUB
    LE, ptMod->CPlus.Val.V_PDOUBLE, ptMod->CMinus.Val.V_PDOUBLE,
    Met->Par[0].Val.V_DOUBLE, Met->Par[1].Val.V_RGDOUBLE,
    Met->Par[2].Val.V_INT,  &(Met->Res[0].Val.V_DOUBLE));

  }

  static int CHK_OPT(AP_CGMY_REALVAR)(void *Opt, void *Mod)
  {
    if ((strcmp( ((Option*)Opt)->Name,"CallRealVarEuro")==0
     )||strcmp( ((Option*)Opt)->Name,"PutRealVarEuro")==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_DOUBLE=10.0;
      Met->Par[1].Val.V_RGDOUBLE=0.5;
      Met->Par[2].Val.V_INT=12;

      first=0;
    }
    return OK;
  }
```

```
  PricingMethod MET(AP_CGMY_REALVAR)=
  {
    "AP_CGMY_REALVAR",
    {   {"Shifting parameter for Laplace transform:",
    DOUBLE,{100},ALLOW    },
        {"Step of discretization for Laplace transform: ",
    RGDOUBLE,{100},ALLOW    },
  {"The log of Nb of points for Laplace transform", INT,{1
    0},ALLOW    },
      {" ",PREMIA_NULLTYPE,{0},FORBID}},
    CALC(AP_CGMY_REALVAR),
    {   {"Price, in annual volatility points",DOUBLE,{100},
    FORBID},
        {" ",PREMIA_NULLTYPE,{0},FORBID}},
    CHK_OPT(AP_CGMY_REALVAR),
    CHK_ok ,
    MET(Init)
  } ;

  /*////////////////////////////////////*/
}
```

References