Help

```
// (c) J. Poirot and P. Tankov, June-September 2006
// Permission granted to redistribute and modify this code
    if the above copyright notice remains intact
// Direct all questions concerning this code to tankov@
    math.jussieu.fr


// Simulation of the CGMY process with Levy measure trunc
    ated at level eps
// (jumps smaller than eps in absolute value are replaced
    with their mean)
// Uses the algorithm in Madan and Yor (), see also Poirot
    and Tankov (2006)
// See header file cgmy.h for explanations

extern "C"{
#include "pnl/pnl_random.h"
#include "pnl/pnl_mathtools.h"
}
#include "cgmy.h"
#include <cmath>
#include <cstdlib>
using namespace std;


namespace {

  class InfExcept{};

  double DegHypergeometric1(double a, double b, double z)
  {
    int i=0;
    double H=1;
    double c=1;
    while(fabs(c)>0.00000001)
      {
        c *= ((a+i)/(b+i))*z/(i+1);
        if(fabs(c)>1e20) throw InfExcept();
        H+=c;
        i++;
        if(i>300) {
          //          std::cout << "Maximum iteration numb
    er reached in DegHypergeometric with z="<<z<<"; H="<<H<<std:
    :endl;
          break;
        }
      }

    return H;
  }

}

namespace{
  double ParCylFunction1 (double p, double z)
  {

    int i;
    double u,v;
    double S=1, c=1;
    double g=tgamma(-p/2+0.5);
    double h=tgamma(-p/2);
    double d;
    if(z<5) /* originally test was z<40, but it creates ov
    erflow in DegHypergeometric1 */
      {
        u= DegHypergeometric1(-p/2,0.5,z*z/2);
        v= DegHypergeometric1(0.5-p/2,1.5,z*z/2);
        d=pow(2,p/2)*exp(-z*z/4)*((sqrt(M_PI)*u/g)-sqrt(2*
    M_PI)*z*v/h);
        return d;
      }
    else{
      for (i=1;i<20;i++)
        {
          c*= -(p-i+1)*(p-i)/(2*i*pow(z,2*i));
          S+=c;}

      return exp(-z*z/4)*pow(z,p)*S;}

  }
  double IntegralI(double Y,double a, double lambda)
  {
    double val;
    val= pow(2*lambda, (-Y)/2)*tgamma(Y)*exp(a*a/(8*lambda)
    )*ParCylFunction1(-Y, (a/sqrt(2*lambda)));

    return val;
  }

  double h(double y, double Y, double A,double B)
  {
    double val1;
    val1=(exp((A*A-B*B)*y/2)*tgamma((Y+1)/2)*pow(2,Y)*pow(
    B*B*y/2,Y/2)*IntegralI(Y,B*B*y,B*B*y/2))/(tgamma(Y)*tgamma(
    0.5));
    return val1;
  }

}

CGMYSimulator::CGMYSimulator(double xC, double xG, double x
    M, double xY, double xeps,int xgenerator) :
  C(xC), G(xG), M(xM), Y(xY), eps(xeps), generator(xgenerator)
{
  P = tgamma(0.5)/(pow(2.0,Y/2)*tgamma((Y+1)/2));
  A = (G-M)/2;
  B = (G+M)/2;
  d = P*C*pow(eps,1-Y/2)/(1-Y/2);
  lambda = 2*C*P/(Y*pow(eps,Y/2));
}

double CGMYSimulator::sim(double t)
{
  double u1, u2, u3, y;
  double H=d*t;
  double K=0.;
  while(K<t)
    {
```

```
   u1=pnl_rand_uni(generator);
   u2=pnl_rand_uni(generator);
   u3=pnl_rand_uni(generator);
   K-=log (u2)/lambda;
   if(K>t) break;
   y= eps * pow(u1,-2./Y);
   try{
    if (h(y,Y,A,B)>u3) H+=y;
   }
   catch(InfExcept){
   }
  }
 u1 = pnl_rand_normal(generator);
 /* printf ("%.16f, %.16f{n", H, u1); */
 return A*H+sqrt(H)*u1;
}

bool CGMYSimulator::simtojump(double & t, double & before,
  double & after)
{
 double y;
 double K=0;
 while(K<t)
  {
   double u1=pnl_rand_uni(generator);
   double u2=pnl_rand_uni(generator);
   double u3=pnl_rand_uni(generator);
   double f=-log (u2)/lambda;
   K += f;
   if(K>t) break;
   y= eps/(pow(u1,2/Y));
   try{
    if (h(y,Y,A,B)>u3){
     t = K;
     double sg1 = pnl_rand_normal(generator);
     double sg2 = pnl_rand_normal(generator);
     before = A*d*t+sqrt(d*t)*sg1;
     after = before + A*y+sqrt(y)*sg2;
     return false;
    }
   }
```

```
      catch(InfExcept){
      }

    }
  double sg = pnl_rand_normal(generator);
  after = before = A*d*t+sqrt(d*t)*sg;
  return true;
}

double CGMYSimulator::cumulant(int n)
{
  if(PNL_IS_EVEN(n)) return C*tgamma(1-Y+n)/(n-Y)*(pow(M,Y-
    n) + pow(G,Y-n));
  else return C*tgamma(1-Y+n)/(n-Y)*(pow(M,Y-n) - pow(G,Y-
    n));
}

double CGMYSimulator::gamma_mart()
{
  if(Y==1)
    return C*(M*log(M)-(M-1)*log(M-1)+G*log(G)-(G+1)*log(G+
    1));
  else
    return C*tgamma(2-Y)*(pow(M,Y)-pow(M-1,Y)+pow(G,Y)-pow(
    G+1,Y))/((Y-1)*Y);
}
```

References