Help

```
#include <iostream>
#include <cmath>
#include <cstdlib>

using namespace std;

#include "
href../../../../common/math/ImportanceSampling_jl/src/MertonModel_h_src.pdfmath/
#include "
href../../../../common/math/jlparser/include/jlparser/parser_h_src.pdfjlparser/p
#include "pnl/pnl_matrix.h"

MertonModel::MertonModel() : JumpModel() { }

MertonModel::~MertonModel()
{
  if (alpha) pnl_vect_free(&alpha);
  if (m) pnl_vect_free(&m);
}

MertonModel::MertonModel(const Param &P)
  : JumpModel(P)
{
  P.extract("log jump mean", m, poissonSize);
  P.extract("log jump variance", alpha, poissonSize);

  /*
   * Compute the drift part of the jump diffusion process
   */
  for (int i = 0 ; i < size ; i++)
    {
      int i_poisson = (poissonSize == 1 ? 0 : i);
      const double sigma_i = GET(sigma, i);
      LET(levyDrift, i) = interest - GET(lambda, i_poisson) *
                          (exp(GET(m, i_poisson) + GET(alpha, i_poisson) / 2.)
                          - GET(dividend, i) - SQR(sigma_i) / 2.0;
    }
  if (poissonSize == size + 1)
    {
```

```
      pnl_vect_minus_double(levyDrift, GET(lambda, size) *
                            (exp(GET(m, size) + GET(alpha, size) / 2.) - 1));
    }
}

/**
 * Computes one path of the model assuming the Brownian increments have
 * already been drawn in Gincr_drift
 * @param rng: PnlRng
 * @param mu is the vector of intensities. They are supposed to be constant
 * for every dimension
 */
void MertonModel::pathMu_aux(PnlRng *rng, const PnlVect *mu)
{
  for (int i = 0 ; i < nTimeSteps ; i++)
    {
      for (int j = 0 ; j < poissonSize ; j++)
        {
          const double alpha_j = GET(alpha, j);
          const double m_j = GET(m, j);
          // Compute number of jumps
          int n = pnl_rng_poisson(GET(mu, j) * dt, rng);
          MLET(poissonMat, i, j) = n;
          // Compute jumps
          double jump = 1.;
          if (n > 0)
            {
              jump = pnl_rng_lognormal(n * m_j, n * n * alpha_j, rng);
            }

          MLET(jumpsMat, i, j) = jump;
        }
    }
  path();
}

/**
 * Auxiliary function.
 *
 * The Gaussian part has to be already drawn and available through
 * mod->Gincr_drift
```

```
 *
 * Compute a path of the Merton model with piecewise constant intensities
 * (given by mu) for the poissonMat processes.
 *
 * @param rng a random number generator
 * @param mu jump intensity (one intensity per time time step)
 */
void MertonModel::pathMuFull_aux(PnlRng *rng, const PnlVect *mu)
{
  PnlMat intensity = pnl_mat_wrap_array(mu->array, nTimeSteps, poissonSize);

  for (int i = 0 ; i < nTimeSteps ; i++)
    {
      for (int j = 0 ; j < poissonSize ; j++)
        {
          const double alpha_j = GET(alpha, j);
          const double m_j = GET(m, j);
          // Compute number of jumps
          int n = pnl_rng_poisson(MGET(&intensity, i, j), rng);
          MLET(poissonMat, i, j) = n;
          // Compute jumps
          double jump = 1.;
          if (n > 0)
            {
              jump = pnl_rng_lognormal(n * m_j, n * n * alpha_j, rng);
            }

          MLET(jumpsMat, i, j) = jump;
        }
    }
  path();
}

void MertonModel::print() const
{
  cout << "**** Merton Model Characteristics ****" << endl;
  cout << " mean : ";
  pnl_vect_print_asrow(m);
  cout << " alpha : ";
  pnl_vect_print_asrow(alpha);
  JumpModel::print();
```

```
}
```