Help

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

using namespace std;

#include "
href../../../../common/math/ImportanceSampling_jl/src/HestonModel_h_src.pdfmath/
#include "
href../../../../common/math/jlparser/include/jlparser/parser_h_src.pdfjlparser/p

HestonModel::HestonModel() :
  StocVolModel(),  reverting_rate(NULL), long_run_var(NULL) { }

HestonModel::~HestonModel()
{
  pnl_vect_free(&reverting_rate);
  pnl_vect_free(&long_run_var);
}

// Do not call the BaseModel constructor because the brownian size is twice the
// size of the model and many objects have to be updated
HestonModel::HestonModel(const Param &P)
    : StocVolModel(P)
{
  P.extract("reverting rate", reverting_rate, size);
  P.extract("long run variance", long_run_var, size);
}

void HestonModel::path()
{
  int i, j;
  double tj;
  // Time 0
  pnl_mat_set_row(pathMatrix, init, 0);
  pnl_vect_clone(sigmaVector, sigma0);

  for (j = 1, tj = dt ; j <= nTimeSteps ; j++, tj += dt)
    {
```

```
      PnlVect G_j = pnl_vect_wrap_mat_row(Gincr_drift, j - 1);
      pnl_mat_mult_vect_inplace(workVector, covChol, &G_j);
      for (i = 0 ; i < size ; i++)
        {
          double sigma_i_plus = max(GET(sigmaVector, i), 0.);
          MLET(pathMatrix, j, i) = MGET(pathMatrix, j - 1, i) * (1. + (interest
           sqrt(sigma_i_plus) * MGET(pathMatrix, j - 1, i) * sqrt_dt * GET(workV
          LET(sigmaVector, i) += GET(reverting_rate, i) * (GET(long_run_var, i)
           + GET(voVol, i) * sqrt(sigma_i_plus) * sqrt_dt * GET(workVector, i +
        }
    }
}

void HestonModel::print() const
{
  cout << endl;
  cout << "**** Heston Model Characteristics ****" << endl;
  cout << " initial volatility "; pnl_vect_print_asrow(sigma0);
  cout << " volatility of volatility "; pnl_vect_print_asrow(voVol);
  cout << " reverting rate "; pnl_vect_print_asrow(reverting_rate);
  cout << " long run variance "; pnl_vect_print_asrow(long_run_var);
  cout << " asset vol correlation " << gamma << endl;
  BaseModel::print();
}
```