Help

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

using namespace std;

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

inline static double sq(double x)
{
  return x * x;
}

DupireModel::DupireModel() : BaseModel() { }

DupireModel::~DupireModel() { }

DupireModel::DupireModel(const Param &P) : BaseModel(P) { }

// /*
//  * Volatility function
//  * dS_t = interest S_t dt + sigma(t, S_t) dW_t
//  */
// double DupireModel::sigma (double maturity, double t, double St, int i)
// {
//   return 0.6*(1.2-exp(-.1*(t))*exp(-0.1e-2 * sq(St*exp(interest*(maturity-t))
//     *exp(-0.5e-1*sqrt(t)) * St;
// }

/*
 * Black Scholes Volatility function for testing purposes
 * dS_t = interest S_t dt + sigma(t, S_t) dW_t
 */
double DupireModel::sigma(double maturity, double t, double St, int i)
{
  return 0.2 * St;
```

```
}

/*
 * Volatility function
 * dS_t = interest S_t dt + sigma(t, S_t) dW_t
 */
// double DupireModel::sigma (double maturity, double t, double St, int i)
// {
//   /* return (0.01 + 0.1*exp(-St/100)+0.01*t)*St; */
//   return 15.0;
// }

void DupireModel::path()
{
  int j;
  double tj;
  // Time 0
  pnl_mat_set_row(pathMatrix, init, 0);

  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 (int i = 0 ; i < size ; i++)
        {
          MLET(pathMatrix, j, i) = MGET(pathMatrix, j - 1, i) *
                                   (1 + (interest - GET(dividend, i)) * dt) +
                                   sqrt_dt * sigma(maturity, tj, MGET(pathMatrix, j
        }
    }
}

void DupireModel::print() const
{
  cout << "**** Local Volatility Model Characteristics ****" << endl;
  BaseModel::print();
}
```