

Help

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

using namespace std;

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

BlackScholesModel::BlackScholesModel() : BaseModel() { }

BlackScholesModel::~BlackScholesModel()
{
    if (sigma)
        pnl_vect_free(&sigma);
}

BlackScholesModel::BlackScholesModel(const Param &P)
    : BaseModel(P)
{
    P.extract("volatility", sigma, size);
}

/**
 * Computes one path of the model using Gincr_drift
 */
void BlackScholesModel::path()
{
    // Time 0
    pnl_mat_set_row(pathMatrix, init, 0);

    pnl_mat_dgemm('N', 'T', 1., covChol, Gincr_drift, 0., workMatrix);
    for (int j = 1 ; j <= nTimeSteps ; j++)
    {
        // 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)
                                * exp((interest - GET(dividend, i) - GET(sigma, i)
                                         * dt + sqrt_dt * GET(sigma, i) * MGET(workM
                                )
        }
    }
}

void BlackScholesModel::print() const
{
    cout << "**** BS Model Characteristics ****" << endl;
    cout << " volatility : ";
    pnl_vect_print_asrow(sigma);
    BaseModel::print();
}

```