Help

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

using namespace std;

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

mcam::BlackScholesModel::BlackScholesModel() : Model(), Cov_Chol(NULL), sigma(NU

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

mcam::BlackScholesModel::BlackScholesModel(const Param &P)
    : Model(P)
{
    brownianSize = size;
    P.extract("volatility", sigma, size);
    P.extract("correlation", rho);
    /* test the value of the correlation */
    if (rho > 1 || rho < -1.0 / (size - 1))
    {
        perror("correlation out of range");
        exit(1);
    }
    Cov_Chol = pnl_mat_create(size, size);
    Rho(rho);
}

/* set the value of the correlation parameter and update the
 * covariance matrix and its derivative */
void mcam::BlackScholesModel::Rho(double rho)
```

```
{
    pnl_mat_set_all(Cov_Chol, rho);
    pnl_mat_set_diag(Cov_Chol, 1., 0);
    pnl_mat_chol(Cov_Chol);
}

/**
 * Computes one path of the model using the normalized Brownian increments on
 * the finer grid
 * @param G matrix of standard normal variables with size ((dates * subticks) x
 */
void mcam::BlackScholesModel::path(const PnlMat *G)
{
    // Time 0
    PnlMat *m = path_m();
    PnlMat *w = work();
    pnl_mat_resize(m, subdates + 1, size);
    pnl_mat_set_row(m, spot, 0);
    pnl_mat_resize(w, subdates, size);
    pnl_mat_dgemm('N', 'T', sqrt_dt, G, Cov_Chol, 0., w);
    for (int j = 0; j < subdates; j++)
    {
        for (int i = 0; i < size; i++)
        {
            MLET(m, j + 1, i) = MGET(m, j, i) * exp((r - GET(dividend, i) - GET(
        }
    }
}

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

}
```

```
PnlVect * mcam::BlackScholesModel::getMin() const
{
    PnlVect *min_v = pnl_vect_create (size);
    for (int i = 0; i < size; i++)
    {
        double sig = GET(sigma, i);
        LET(min_v, i) = GET(spot,i) * exp (MIN(r - sig*sig/2, 0) * T - sig * 4.0
    }
    return min_v;
}

PnlVect * mcam::BlackScholesModel::getMax() const
{
    PnlVect *max_v = pnl_vect_create (size);
    for (int i = 0; i < size; i++)
    {
        double sig = GET(sigma, i);
        LET(max_v, i) = GET(spot,i) * exp (MAX(r - sig*sig/2, 0) * T + sig * 4.0
    }
    return max_v;
}
```