

[Help](#)

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

using namespace std;

#include "math/ImportanceSampling_jl/src/SABRModel.hpp"
#include "math/ImportanceSampling_jl/src/parser.hpp"

SABRModel::SABRModel() :
    StocVolModel(), beta(NULL) { }

SABRModel::~SABRModel()
{
    pnl_vect_free(&beta);
}

// Do not call the BaseModel constructor because the brownian size is twice the
// size of the model and many objects have to be updated
SABRModel::SABRModel(const Param &P)
    : StocVolModel(P)
{
    P.extract("sabr exponent", beta, size);
}

void SABRModel::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++)
        {
```

```

        MLET(pathMatrix, j, i) =
            MGET(pathMatrix, j - 1, i) * (1. + (interest - GET(dividend, i)) * dt
            sqrt_dt * GET(sigmaVector, i) * pow(MGET(pathMatrix, j - 1, i), GET(b
            LET(sigmaVector, i) *= (1. + GET(voVol, i) * sqrt_dt * GET(workVector,
    }
}

void SABRModel::print() const
{
    cout << endl;
    cout << "**** SABR Model Characteristics ****" << endl;
    cout << " initial volatility "; pnl_vect_print_asrow(sigma0);
    cout << " volatility of volatility "; pnl_vect_print_asrow(voVol);
    cout << " exponent "; pnl_vect_print_asrow(beta);
    cout << " asset vol correlation " << gamma << endl;
    BaseModel::print();
}

```