

[Help](#)

```
#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2007+2) //The "#els
#else

/// \ file gm.h
/// \ brief gaussian mapping technique
/// \ author M. Ciuca (MathFi, ENPC)
/// \ note (C) Copyright Premia 8 - 2006, under Premia 8 Software license
//
// Use, modification and distribution are subject to the
// Premia 8 Software license

//#define NDEBUG
#include <cassert>

#ifndef _GM_H
#define _GM_H

// The couple of files (gm.h, gm.cpp) implements the Gaussian Mapping
// approximation technique presented in the paper:
// Brigo D., Alfonsi A. (2004), "Credit Default Swaps calibration and option
// pricing with the SSRD stochastic intensity and interest-rate model"

#include <stdexcept>
#include <iostream>
#include <fstream>
#include <iomanip>
#include <string>
#include <
href../../../../common/math/highdim_solver/highdim_vector_h_src.pdfvector>

#include "
href../../../../common/math/credit_cds/base_h_src.pdfbase.h"
#include "
href../../../../common/math/credit_cds/cdscirpp_h_src.pdfcirpp.h"

class CIR_Mapped_toVasicek
```

```

{
public:
    CIR_Mapped_toVasicek(double k = 0, double theta = 0, double sigma = 0, double
                        double T = 0);

    double Compute_ZC_CIR(double T);
    double Compute_ZC_CIR_d(double T);
    double Compute_ZC_CIR_d_num(double T);
    double Compute_ZC_CIRn(double T);
    double Compute_ZC_Vasicek(double T);
    double Compute_ZC_Vasicekn(double T);
    double Compute_VasicekMappedVolatility(double T);

private:
    double _k;
    double _theta;
    double _sigma;
    double _sigma_v;
    double _x0;
    double _T;

    double _h;

    double A_Vasicek(double t);
    double A_CIR(double t);
    inline double A_CIR_d(double t);
    inline double H_CIR(double t);
    inline double H_CIR_d(double t);
    inline double H_CIR_n(double t);
    inline double H_CIR_n_d(double t);
    double B_CIR(double t);
    inline double B_CIR_deriv(double t);
    inline double B_CIR_n(double t);
    inline double B_CIR_n_d(double t);
    inline double B_CIR_d(double t);
    inline double B_CIR_d_d(double t);

    friend double g(double k, double T);
};

```

```

class CDS_GaussianMapping_Old
{
public:
    CDS_GaussianMapping_Old(double k, double theta, double sigma, double x0,
                            string inputCDS,
                            double k_r, double theta_r, double sigma_r, double x0_r,
                            string inputShortRate,
                            double rho,
                            std::vector<double> &timesT, double Z);
    CDS_GaussianMapping_Old(double k, double theta, double sigma, double x0,
                            std::vector<double> &intensityMat,
                            std::vector<double> &intensityRates,
                            double k_r, double theta_r, double sigma_r, double x0_r,
                            std::vector<double> &RatesMat,
                            std::vector<double> &Rates,
                            double rho,
                            std::vector<double> &timesT, double Z);

    void Get_sigmas(double t);

    double Quote(double T, int noTi, double &defaultLeg, double &paymentLeg);
    double CDS(double Rf);

    double Compute_M2_Vasicek(double T);
    double Compute_M2_Vasicek_Deg(double T);
    double Compute_M1_Vasicek(double T);
    double Compute_M1_Vasicek_Deg(double T);
    double Compute_M1_Vasicek_Corrected(double T);
    double Compute_M1_Vasicek_Corrected_num(double T);

    double sigma_v(double t)
    {
        return _intensity.Compute_VasicekMappedVolatility(t);
    }
    double sigma_v_r(double t)
    {
        return _shortRate.Compute_VasicekMappedVolatility(t);
    }

    void reset()
    {
        I1 = I2 = 0;
    }
}

```

```

}

private:
    double _k;
    double _theta;
    double _sigma;
    double _x0;
    double _k_r;
    double _theta_r;
    double _sigma_r;
    double _x0_r;
    std::vector<double> _timesT;
    double _Z;

    CIRppDI _cirpp_intensity;
    CIRppSR _cirpp_shortRate;

    CIR_Mapped_toVasicek _shortRate;
    CIR_Mapped_toVasicek _intensity;
    double _rho;
    double _T;

    double I1;
    double I2;
    double S;

    double mean_A(double t);
    double mean_Ad(double t);
    double mean_B(double t);
    double variance_A(double t);
    double variance_Ad(double t);
    double variance_B(double t);
    double rho_bar(double t);
    double rho_bard(double t);

    double Delta(double t);
    double Delta_num(double t);

    friend double g(double k, double T);

```

```

double f1(double u);
double f2(double u);
double f_Sum(int n0, int n);
double sum_of_shifts(double u)
{
    return _cirpp_intensity.Phi(u) + _cirpp_shortRate.Phi(u);
}

NumInt<CDS_GaussianMapping_Old> numInt;
};

#endif
#endif //PremiaCurrentVersion

```