

[Help](#)

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#include "pnl/pnl_vector.h"
#include "pnl/pnl_matrix.h"
#include "pnl/pnl_mathtools.h"
#include "pnl/pnl_interpolation.h"
#include "pnl/pnl_integration.h"

#include "lmm_stochvol_piterbarg.h"

static int _n_swap;
static int _m_swap;

double ParametricForm(PnlMat *Params, double t, double Tn, int k)
{
    return (MGET(Params, 0, k) * (Tn - t) + MGET(Params, 1, k)) * exp(-MGET(Params, 2, k) * (Tn - t));
}

// Libor instantaneous volatility functions.
// t: time. Tn:maturity of Libor. k: index of the diffusion factor
double LiborRate_vol(StructLmmPiterbarg *LmmPiterbarg, double t, double Tn, int k)
{
    if (t <= Tn)
    {
        return ParametricForm(LmmPiterbarg->VolsParams, t, Tn, k);
    }
    else return 0.;
}

// Libor instantaneous skew functions.
double LiborRate_skew(StructLmmPiterbarg *LmmPiterbarg, double t, double Tn)
{
    if (t <= Tn) return ParametricForm(LmmPiterbarg->SkewsParams, t, Tn, 0);
    else return 0.;
}
```

```

// This function create a structure StructLmmPiterbarg that contains info about
StructLmmPiterbarg *SetLmmPiterbarg(int InitYieldCurve_flag, double R_flat, char
{
    int N = pnl_iround(T_last / period) - 1;
    int i;
    StructLmmPiterbarg *LmmPiterbarg = malloc(sizeof(StructLmmPiterbarg));

    LmmPiterbarg->ZCMarket = malloc(sizeof(ZCMarketData));
    LmmPiterbarg->ZCMarket->filename = curve;
    SetInitYieldCurve(InitYieldCurve_flag, R_flat, LmmPiterbarg->ZCMarket);

    LmmPiterbarg->TimeDates = pnl_vect_create(N + 1);

    LmmPiterbarg->SkewsParams = pnl_mat_new();
    pnl_mat_clone(LmmPiterbarg->SkewsParams, SkewsParams);
    LmmPiterbarg->VolsParams = pnl_mat_new();
    pnl_mat_clone(LmmPiterbarg->VolsParams, VolsParams);

    LmmPiterbarg->NbrVolFactors = NbrVolFactors;

    LmmPiterbarg->Var_SpeedMeanReversion = Var_SpeedMeanReversion;
    LmmPiterbarg->Var_Volatility = Var_Volatility;

    for (i = 0; i <= N; i++) LET(LmmPiterbarg->TimeDates, i) = (i + 1) * period;

    return LmmPiterbarg;
}

// Free StructLmmPiterbarg
void FreeLmmPiterbarg(StructLmmPiterbarg **LmmPiterbarg)
{
    pnl_vect_free(&((*LmmPiterbarg)->TimeDates));

    DeleteZCMarketData((*LmmPiterbarg)->ZCMarket);
    free((*LmmPiterbarg)->ZCMarket);

    pnl_mat_free(&((*LmmPiterbarg)->SkewsParams));
    pnl_mat_free(&((*LmmPiterbarg)->VolsParams));

    free(*LmmPiterbarg);
    LmmPiterbarg = NULL;
}

```

```
}

```

```
// T_{i-1} < s <= T_{i}
int indiceTimeGrid(PnlVect *TimeGrid, double s)
{
    int i = 0, N = TimeGrid->size - 1;

    while (i <= N && s - GET(TimeGrid, i) > 1e-10) i++;

    return i;
}

```

```
double SwapRate_vol_k(StructLmmPiterbarg *LmmPiterbarg, double t, int n_swap, int m_swap)
{
    int i;
    double Ti1, Ti2, Tn, Tm, P0_Tn, P0_Tm, q_i_n_m, swp_vol_k_n_m;

    Tn = GET(LmmPiterbarg->TimeDates, n_swap);
    Tm = GET(LmmPiterbarg->TimeDates, m_swap);

    P0_Tn = BondPrice(Tn, LmmPiterbarg->ZCMarket);
    P0_Tm = BondPrice(Tm, LmmPiterbarg->ZCMarket);

    swp_vol_k_n_m = 0.;
    for (i = n_swap; i < m_swap; i++)
    {
        Ti1 = GET(LmmPiterbarg->TimeDates, i);
        Ti2 = GET(LmmPiterbarg->TimeDates, i + 1);
        q_i_n_m = BondPrice(Ti1, LmmPiterbarg->ZCMarket) - BondPrice(Ti2, LmmPiterbarg->ZCMarket);

        swp_vol_k_n_m += q_i_n_m * LiborRate_vol(LmmPiterbarg, t, Ti1, k);
    }
    swp_vol_k_n_m /= (P0_Tn - P0_Tm);

    return swp_vol_k_n_m;
}

```

```
double SwapRate_vol(StructLmmPiterbarg *LmmPiterbarg, double t, int n_swap, int m_swap, int k)
{
    int k;

```

```

double swp_vol_n_m = 0.;

for (k = 0; k < LmmPiterbarg->NbrVolFactors; k++)
{
    swp_vol_n_m += pow(SwapRate_vol_k(LmmPiterbarg, t, n_swap, m_swap, k), 2);
}

return sqrt(swp_vol_n_m);
}

double SwapRate_skew(StructLmmPiterbarg *LmmPiterbarg, double t, int n_swap, int
{
    int i, k;
    double Ti, sum_vol_skew, swp_skew_n_m, sqr_swp_vol_n_m, swp_vol_k_n_m;

    swp_skew_n_m = 0.;
    sqr_swp_vol_n_m = 0.;

    for (k = 0; k < LmmPiterbarg->NbrVolFactors; k++)
    {
        swp_vol_k_n_m = SwapRate_vol_k(LmmPiterbarg, t, n_swap, m_swap, k);
        sqr_swp_vol_n_m += pow(swp_vol_k_n_m, 2);

        sum_vol_skew = 0.;
        for (i = n_swap; i < m_swap; i++)
        {
            Ti = GET(LmmPiterbarg->TimeDates, i);
            sum_vol_skew += LiborRate_vol(LmmPiterbarg, t, Ti, k) * LiborRate_skew
        }

        swp_skew_n_m += swp_vol_k_n_m * sum_vol_skew;
    }

    return swp_skew_n_m / (sqr_swp_vol_n_m * (m_swap - n_swap));
}

static double func_to_intg_1(double t, void *LmmPiterbarg)
{
    double tmp = SwapRate_vol(LmmPiterbarg, t, _n_swap, _m_swap);
    return tmp * tmp;
}

```

```

}

static double func_to_intg_2(double s, void *LmmPiterbarg)
{
    double theta = ((StructLmmPiterbarg *)LmmPiterbarg)->Var_SpeedMeanReversion;
    return pow(SwapRate_vol(LmmPiterbarg, s, _n_swap, _m_swap), 2) * (exp(theta *
}

static double func_to_intg_3(double t, void *LmmPiterbarg)
{
    PnlFunc func;
    int NbrPts = 20;
    double v1, v2, v_nm2;
    double theta = ((StructLmmPiterbarg *)LmmPiterbarg)->Var_SpeedMeanReversion, e

    func.params = LmmPiterbarg;
    func.F = func_to_intg_1;
    v1 = pnl_integration(&func, 0.0, t, NbrPts, "simpson");

    func.F = func_to_intg_2;
    v2 = pnl_integration(&func, 0.0, t, NbrPts, "simpson");

    v_nm2 = v1 + SQR(eta) * exp(-theta * t) * v2;

    return v_nm2 * pow(SwapRate_vol(LmmPiterbarg, t, _n_swap, _m_swap), 2);
}

static double func_to_intg_4(double t, void *LmmPiterbarg)
{
    return func_to_intg_3(t, LmmPiterbarg) * SwapRate_skew(LmmPiterbarg, t, _n_swa
}

double SwapRate_skew_avg(StructLmmPiterbarg *LmmPiterbarg, int n_swap, int m_swa
{
    PnlFunc func;
    int NbrPts = 30;
    double result, sumw, Tn = GET(LmmPiterbarg->TimeDates, n_swap);

    _n_swap = n_swap;
    _m_swap = m_swap;

```

```

func.params = LmmPiterbarg;
func.F = func_to_intg_4;
result = pnl_integration(&func, 0.0, Tn, NbrPts, "simpson");

func.F = func_to_intg_3;
sumw = pnl_integration(&func, 0.0, Tn, NbrPts, "simpson");

return result / sumw;
}

static double log_LapTransf_intg_v0(StructLmmPiterbarg *LmmPiterbarg, double u,
{
    double Tn, gamma, exp_g_Tn, B_0, A_0;
    double theta = LmmPiterbarg->Var_SpeedMeanReversion, eta = LmmPiterbarg->Var_V

    Tn = GET(LmmPiterbarg->TimeDates, n_swap);
    gamma = sqrt(SQR(theta) + 2 * SQR(eta) * u);
    exp_g_Tn = exp(-gamma * Tn);

    B_0 = 2 * u * (1 - exp_g_Tn) / ((theta + gamma) * (1 - exp_g_Tn) + 2 * gamma *
    A_0 = 2 * theta / SQR(eta) * log(2 * gamma / ((theta + gamma) * (1 - exp_g_Tn)

    return (A_0 - B_0);
}

static double LapTransf_F(StructLmmPiterbarg *LmmPiterbarg, double t, double y,
{
    double theta = LmmPiterbarg->Var_SpeedMeanReversion, eta = LmmPiterbarg->Var_V

    return -theta * y - 0.5 * SQR(eta * y) + u * pow(SwapRate_vol(LmmPiterbarg, t,
}

static double log_LapTransf_intg_v1(StructLmmPiterbarg *LmmPiterbarg, double u,
{
    double B_i, A_i, h, ti, k1, k2, k3, k4;
    int i, n_step = 2 * n_swap;
    double theta = LmmPiterbarg->Var_SpeedMeanReversion;

    B_i = 0.;
    A_i = 0.;

```

```

ti = 0.;
h = GET(LmmPiterbarg->TimeDates, n_swap) / n_step;

for (i = 0; i < n_step; i++)
{
    k1 = LapTransf_F(LmmPiterbarg, ti, B_i, u, n_swap, m_swap);
    k2 = LapTransf_F(LmmPiterbarg, ti + 0.5 * h, B_i + 0.5 * h * k1, u, n_swap);
    k3 = LapTransf_F(LmmPiterbarg, ti + 0.5 * h, B_i + 0.5 * h * k2, u, n_swap);
    k4 = LapTransf_F(LmmPiterbarg, ti + h, B_i + h * k3, u, n_swap, m_swap);
    B_i = B_i + h / 6.*(k1 + 2 * k2 + 2 * k3 + k4); // Runge-Kutta of order 4

    A_i -= theta * B_i * h;
    ti += h;
}

A_i += 0.5 * theta * B_i * h;

return (A_i - B_i);
}

double Func_zero(StructLmmPiterbarg *LmmPiterbarg, double lambda, int n_swap, int m_swap)
{
    return log_LapTransf_intg_v0(LmmPiterbarg, c * SQR(lambda), n_swap, m_swap) -
}

double SwapRate_vol_avg(StructLmmPiterbarg *LmmPiterbarg, int n_swap, int m_swap)
{
    int i;
    double c, phi_c, intg_sigma_n_m, lambda, lambda_sup = 0., lambda_inf = 0., f_l;
    double Tn = GET(LmmPiterbarg->TimeDates, n_swap), precision = 1e-7;
    PnlFunc func;
    int NbrPts = 2 * n_swap;

    _n_swap = n_swap;
    _m_swap = m_swap;

    func.params = LmmPiterbarg;
    func.F = func_to_intg_1;
    intg_sigma_n_m = pnl_integration(&func, 0.0, Tn, NbrPts, "simpson");

    c = (4. + SQR(skew_n_m) * intg_sigma_n_m) / (8.*intg_sigma_n_m);

```

```
phi_c = log_LapTransf_intg_v1(LmmPiterbarg, c, n_swap, m_swap);

if (LmmPiterbarg->Var_Volatility == 0) return sqrt(intg_sigma_n_m / Tn);

for (i = 0; i <= n_swap; i++)
    lambda_inf = MAX(lambda_inf, SwapRate_vol(LmmPiterbarg, GET(LmmPiterbarg->Ti

i = 0;
do
{
    lambda = 0.5 * (lambda_inf + lambda_sup);
    f_lambda = Func_zero(LmmPiterbarg, lambda, n_swap, m_swap, skew_n_m, c, ph

    if (f_lambda < 0) lambda_inf = lambda;
    else lambda_sup = lambda;
    i++;
}
while (fabs(f_lambda) > precision && fabs(lambda_sup - lambda_inf) > precision

return lambda;
}
```