

[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"
#include "ap_averagingtech_lmmpit.h"

static double u_st;
static int FlagClosedFormula;
static int n_swap;
static int m_swap;
static double T;
static double SpeedReversionVar;
static double LongTermVar;
static double InitialVar;
static double VolVar;
static double log_Dzeta_Kstar;

static double CharactFunc_CstVol(double u, StructLmmPiterbarg *LmmPiterbarg)
{
    double alpha = 0.5 * (SQR(u) + 0.25);
    double gamma = 0.5 * SQR(VolVar);
    double d = sqrt(SQR(SpeedReversionVar) + 4 * alpha * gamma);
    double exp_d_T = exp(-d * T);
    double g = (SpeedReversionVar - d) / (SpeedReversionVar + d);

    double B = (SpeedReversionVar - d) / SQR(VolVar) * (1 - exp_d_T) / (1 - g * ex

    double A = SpeedReversionVar * LongTermVar / SQR(VolVar) * ((SpeedReversionVar

    return exp(A + B * InitialVar);
}
```

```
void Runge_Kutta_step(int neqn, double t, const double *y, double *yp, void *Lmm)
{
    *yp = -SpeedReversionVar * (*y) - 0.5 * (SQR(u_st) + 0.25) * pow(SwapRate_vol(
```

```
double CharactFunc_TimeDep_RK4(double u, StructLmmPiterbarg *LmmPiterbarg)
{
    double abserr, A_i;
    int flag;
    int i;
    int n_step;
    double relerr;
    double t, dt;
    double t_out;
    double t_start;
    double t_stop;
    double B_i;
    PnlODEFunc f;

    u_st = u;
    abserr = sqrt(1E-9);
    relerr = sqrt(1E-9);

    flag = 1;

    t_start = 0.0;
    t_stop = T;

    n_step = 150;

    f.F = Runge_Kutta_step;
    f.neqn = 1;
    f.params = LmmPiterbarg;

    t = 0.0;
    t_out = 0.0;
    B_i = 0.0;
    A_i = 0.0;
    dt = (t_stop - t_start) / ((double)n_step);

    for (i = 0 ; i < n_step ; i++)
```

```

    {
        t_out = t + dt;
        pnl_ode_rkf45(&f, &B_i, t, t_out, relerr, abserr, &flag);
        A_i += dt * B_i;
    }

    A_i *= SpeedReversionVar * LongTermVar;

    return exp(A_i + B_i * InitialVar);
}
/*****

double func_to_intg(double u, void *LmmPiterbarg)
{
    double phi;

    if (FlagClosedFormula == 0) phi = CharactFunc_CstVol(u, LmmPiterbarg);
    else phi = CharactFunc_TimeDep_RK4(u, LmmPiterbarg);

    return cos(u * log_Dzeta_Kstar) * phi / (u * u + 0.25);
}

// Call_Put=1 -> Call
// Call_Put=-1 -> Put
double cf_displaced_heston(double S_in, double K_in, double T_in, double Shift_in)
{
    double result_integral = 0., abserr, K_star, option_price;
    int neval;
    PnlFunc func;

    if (FlagClosedFormula == 1) VolConst_in = 1;

    if (Shift_in < 0)
    {
        Call_Put = -Call_Put;
    }

    T = T_in;
    SpeedReversionVar = SpeedReversionVar_in;

```

```

LongTermVar = SQR(Shift_in * VolConst_in) * LongTermVar_in;
InitialVar = SQR(Shift_in * VolConst_in) * InitialVar_in;
VolVar = fabs(Shift_in) * VolConst_in * VolVar_in;
K_star = Shift_in * K_in + (1 - Shift_in) * S_in;
log_Dzeta_Kstar = log(S_in / K_star);

func.params = LmmPiterbarg;
func.F = func_to_intg;
pnl_integration_qag(&func, 0., PNL_POSINF, 1e-5, 1e-5, 1000, &result_integral,

option_price = S_in * (1. - sqrt(K_star / S_in) * result_integral / M_PI);

if (Call_Put == 1) return option_price / fabs(Shift_in);
else return (option_price - S_in + K_star) / fabs(Shift_in);
}

// Payer_Receiver=1 -> Payer
// Payer_Receiver=-1 -> Receiver
double cf_lmm_stochvol_piterbarg_swpt(StructLmmPiterbarg *LmmPiterbarg, double Tn, double Tm)
{
    double skew_avg, vol_avg, swap_rate, swaption_price, swap_numeraire, T_i;
    int i;

    FlagClosedFormula = FlagClosedFormula_in;
    n_swap = indiceTimeGrid(LmmPiterbarg->TimeDates, Tn);
    m_swap = indiceTimeGrid(LmmPiterbarg->TimeDates, Tm);

    swap_rate = ATMSwaptionStrike(Tn, Tm, period, LmmPiterbarg->ZCMarket);
    skew_avg = SwapRate_skew_avg(LmmPiterbarg, n_swap, m_swap);
    vol_avg = SwapRate_vol_avg(LmmPiterbarg, n_swap, m_swap, skew_avg);

    swap_numeraire = 0.;
    T_i = Tn;
    for (i = 0; i < m_swap - n_swap; i++)
    {
        T_i += period;
        swap_numeraire += BondPrice(T_i, LmmPiterbarg->ZCMarket);
    }
    swap_numeraire *= period;
}

```

```

    swaption_price = cf_displaced_heston(swap_rate, swaption_strike, Tn, skew_avg,

    swaption_price *= Nominal * swap_numeraire;

    return swaption_price;
}

// flag_caplfloor=1 -> cap
// flag_caplfloor=-1 -> floor
double cf_lmm_stochvol_piterbarg_capfloor(StructLmmPiterbarg *LmmPiterbarg, double
{
    double skew_avg = 0., vol_avg = 0., libor_rate, caplet_price = 0., cap_price,
    double P1, P2;
    int i, nbr_payments = pnl_iround((Tm - Tn) / period);

    FlagClosedFormula = FlagClosedFormula_in;
    n_swap = indiceTimeGrid(LmmPiterbarg->TimeDates, Tn);
    cap_price = 0.;
    for (i = 0; i < nbr_payments; i++)
    {
        m_swap = n_swap + 1;

        P1 = BondPrice(Tn + i * period, LmmPiterbarg->ZCMarket);
        P2 = BondPrice(Tn + (i + 1) * period, LmmPiterbarg->ZCMarket);
        numeraire = period * P2;

        libor_rate = (P1 / P2 - 1.) / period;
        skew_avg = SwapRate_skew_avg(LmmPiterbarg, n_swap, m_swap);
        vol_avg = SwapRate_vol_avg(LmmPiterbarg, n_swap, m_swap, skew_avg);

        caplet_price = Nominal * numeraire * cf_displaced_heston(libor_rate, cap_s

        cap_price += caplet_price;
        n_swap++;
    }

    return caplet_price;
}

```