

[Help](#)

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

/*-----*/
/*  CF approx. for caplet prices in one-factor LMM with jumps */
/*  Algorithm of Glasserman/Merener */
/* */
/*-----*/
/*  Sonke Blunck, Premia 2005 */
/*-----*/

#ifndef LMM_JUMP_CAPLETPRICE_GLASSMER_H
#define LMM_JUMP_CAPLETPRICE_GLASSMER_H

#include <valarray>

int lmm_jump_caplet_GlassMer_pricer(double tenor, double capletMat, double K, do
// caplet pricing via the CF approx. method of Glassermann/Merener
int lmm_jump_caplet_MC_pricer(double tenor, double capletMat, double K, double f
// caplet pricing via Monte Carlo

class GlassMer // Glasserman/Merener
// one-factor LIBOR Market Model with jumps
{
    const double _delta; // accrual period
    const double _gamma; // diffusion coefficient
    const double _h; // MC time step size

    double _sqrt_h; // square root of _h
    double _t; // current time
    double _Xi; // sum of indep. exponential rv's
    double _psi_factor; // for the fct. psi
    double _x0, _x1; // limits of integr. for the lognormal density
    double _DeltaX; // length of discret. steps for integr. the
    // lognormal density
```

```

int _a_ctr_max;

int _M;           // (number of tenor dates) -1
int _eta;         // index of current accrual period
int _eta_old;     // value of _eta at preceding time step
int _a_ctr;       //

std::valarray<double> _T;           // tenor dates
std::valarray<double> _L0;          // initial LIBOR values
std::valarray<double> _Lt;          // current LIBOR values
std::valarray<double> _lambda;      // jump intensity
std::valarray<double> _sigma;       // parameter of the fct H in the doc.
std::valarray<double> _DeltaJ;      // jumps
std::valarray<double> _a;           // for the forward measure drift a
std::valarray<double> _H;           // for the result of the fct H

public:
    GlassMer(double delta = 0.5, double L0 = 0.06, double gamma = 0.1,
              double h = 0.01, int M = 4);

    void InitialCond(int generator);

    int eta(double t);
    // returns the index k such that t is in (_T[k-1],_T[k]]

    void Set_t(double t);

    double H(int i, double x, double t);

    void Set_H(double x);

    double phi(int i);
    // returns phi_i(_t,_H,_Lt) as in the documentation

    double psi(int i);
    // returns psi_i(_t,_Lt) as in the documentation

    double a(int i);
    // returns the forward measure drift  $a^i_{_t}$  as in the docum.

```

```

void Set_a(int i0);

double Lambda(double t);

void Scheme(int generator);
// one simulation step (from _t to _t+_h) under the spot measure

double CapletMC(double K, int M, int generator);
// MC simulation of the spot measure dynamics

double CapletCF(double K = 0.07);
// the Glassermann/Merener CF approximation

}; // end of the class GlassMer

#endif
#endif //PremiaCurrentVersion

```