```
   Help
#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
     (2007+2) //The "#else" part of the code will be freely av
    ailable after the (year of creation of this file + 2)
#else

/// {file cdscirppmc.h
/// {brief CDS_CIRpp_MC class
/// {author  M. Ciuca (MathFi, ENPC)
/// {note (C) Copyright Premia 8 - 2006, under Premia 8 Sof
    tware license
//
//  Use, modification and distribution are subject to the
//  Premia 8 Software license

#ifndef _CDS_CIRPP_MC_H
#define _CDS_CIRPP_MC_H

#include "cirpp.h"

// This class is made to compute Rf, the CDS Rate, by
    Monte-Carlo.
// The default intensity and interest rates may be
    correlated.
class CDS_CIRpp_MC
{
public:
  CDS_CIRpp_MC(int generator, double k, double theta,
    double sigma, double x0, double barrier,
    string inputCDS,
    double k_r, double theta_r, double sigma_r, double x0
    _r,
    string inputShortRate,
    double rho,
    vector<double>& timesT, double Z,
    double precision = 1.e-04,
    int noTau_Sim=10000
    ):
  _tau( generator, k, theta, sigma, x0,
    timesT[timesT.size()-1], barrier, inputCDS, precision
    ),
```

```
  _shortRate( generator, k_r, theta_r, sigma_r, x0_r,
  timesT[timesT.size()-1], rho, inputShortRate, precisi
  on),
  _timesT(timesT),
  _Z(Z),
  _noTau_Sim(noTau_Sim),
  _b(0.),
  _c(0.)
{
  std::cout << "nMc : " << _noTau_Sim << endl;
}

CDS_CIRpp_MC(int generator, double mrIntensity, double
  thetaIntensity,
  double sigmaIntensity, double y0,
  vector<double>& intensityMat,
  vector<double>& intensityRates,
  double mrRate, double thetaRate, double sigmaRate,
  double x0_r,
  vector<double>& RatesMat,
  vector<double>& Rates,
  double correlation, double maturity, double period,
  double recovery,
  int Nsim,
  double precision = 1.e-04,
  double barrier = 1.0
  );

void WriteCharacteristics();

double CdsRate();
double CdsRate(double& DefaultLeg, double& PaymentLeg,
  double& std_dev_DefaultLeg, double& std_dev_Payment
  Leg);
int MonteCarlo(double& sumI1, double& sumI2, double& su
  mS, int nS=100);
int MonteCarlo(double& DefaultLeg, double& PaymentLeg,
  double& std_dev_DefaultLeg, double& std_dev_Payment
  Leg, int nS=100);
double CdsRate_ControlVariate(double& DefaultLeg,
  double& PaymentLeg,
```

```
    double& std_dev_DefaultLeg, double& std_dev_Payment
    Leg);
  double CdsRate_ControlVariate();

  void Compute_b_and_c(double& b, double& c, double meanI,
     double meanJ, double meanS, int N)
  {
    Estimate_b_and_c(meanI, meanJ, meanS, N);
    //std::cout << "b: " << _b << ", c: " << _c << endl;
    b=_b; c=_c;
  }
  void Set_b_and_c(double b, double c)
  { _b = b; _c = c; }

  double DefaultableZC_MC(double t);
  double DefaultableZC_Mkt(double t);

protected:
  DefaultTimeCIRpp _tau;
  CIRppSR_Explicit0_Correlated _shortRate;
  vector<double> _timesT;
  double _Z;
  int _noTau_Sim;

private:
  double _b;
  double _c;
  bool Generate_Yi(double& sumI1, double& sumI2, double&
    sumS, bool& reset);
  bool Generate_Yi(double& sumI1, double& sumI2, double&
    sumS,
    double& sumI1_sqr, double& sumI2_sqr, double& sumS_sq
    r, bool& reset);
  void Estimate_b_and_c(double meanI, double meanJ,
    double meanS, int N);
};

#endif

#endif //PremiaCurrentVersion
```

References