

## [Help](#)

```
extern "C" {
#include "
href../../mod/cirpp2d/cirpp2d_stdh/cirpp2d_stdh_src.pdfcirpp2d_stdh.h"
#include "
href../../common/enums_h_src.pdfenums.h"
    extern char premia_data_dir[MAX_PATH_LEN];
    extern char *path_sep;
}

#include <
href../../common/math/highdim_solver/highdim_vector_h_src.pdfvector>
#include <fstream>
#include <algorithm>
#include <iterator>
#include <exception>
#include <cmath>
#include "
href../../common/math/credit_cds/cdscirpp_h_src.pdfmath/credit_cds/cdscirpp.h

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2007+2) //The "#els
#else
static int mc_cirpp(
    int flag_data,
    double date,
    double x0_r,
    double mrRate,
    double thetaRate,
    double sigmaRate,
    double x0,
    double mrIntensity,
    double thetaIntensity,
    double sigmaIntensity,
    double correlation,
    double maturity,
    int period,
    double recovery,
    int nsim,
    double precision,
    double barrier,
```

```

int generator,
double *spread,
double *spread_stddev)
{
    maturity -= date;

    std::string path(premia_data_dir);
    path += path_sep;

    std::ifstream zcb((path + "zcb.txt").c_str());

    if (!zcb)
        return UNABLE_TO_OPEN_FILE;

    double T, P;

    std::vector<double>    RatesMat, Rates;
    std::vector<double>    intMat, intRates;

    while (zcb >> T >> P)
    {
        RatesMat.push_back(T);
        Rates.push_back(P);
    }

    if (flag_data == 0)
    {
        std::ifstream intensity_file((path + "intensity.txt").c_str());

        if (!intensity_file)
            return UNABLE_TO_OPEN_FILE;

        while (intensity_file >> T >> P)
        {
            intMat.push_back(T);
            intRates.push_back(P);
        }
    }
    else
    {
        std::ifstream cds_file((path + "cds.txt").c_str());
    }
}

```

```

if (!cds_file)
    return UNABLE_TO_OPEN_FILE;

std::vector<double> spreadMat, spreads;

while (cds_file >> T >> P)
{
    spreadMat.push_back(T);
    spreads.push_back(P);
}

// TODO: put it to parameters.
// What's to do with recovery and period?
double r = 0.03;

DefaultIntensityCalibration(recovery, period, spreadMat, spreads, r, intMa
}

int simulation_dim = 1;

/*MC sampling*/
pnl_rand_init(generator, simulation_dim, 1);

double dl, pl;
double stddev_DL;
double stddev_PL;

/*Price*/
*spread = cds_spread_CIRPPMC_CV(// Computes the value of the spread which corr
    maturity, // maturity of the CDS (in years)
    period, // payment period, in months
    recovery, // expected recovery rate
    precision, // time step for CIR processes path simulation scheme
    nsim, // number of Monte Carlo simulations
    mrRate, // mean reversion coefficient in the interest rate model
    mrIntensity, // mean reversion coefficient in the intensity model
    sigmaRate, // volatility coefficient in the interest rate model
    sigmaIntensity, // volatility coefficient in the intensity model

```

```

        thetaRate, // long-run mean in the interest rate model
        thetaIntensity, // long-run mean in the intensity model
        x0_r, // Starting value of the short rate process
        x0, // Starting value of the intensity process
        correlation, // correlation between rate and intensity
        RatesMat, // Maturities of zero-coupons for calibration
        Rates, // rates of risk-free zero-coupons for calibration
        intMat, // Maturities of CDS used for calibration
        intrRates, // intensity of the name underlying the CDS; (spreads of
        dl, // DefaultLeg price (return parameter)
        pl, // PaymentLeg price (return parameter)
        stddev_DL, // DefaultLeg standard deviation (return parameter)
        stddev_PL, // PaymentLeg standard deviation (return parameter)
        barrier, // Barrier for the intensity process

        generator
    ) ;

    *spread_stddev = *spread * (stddev_DL / dl + stddev_PL / pl) / sqrt(1.0 * nsim);

    return OK;
}
#endif //PremiaCurrentVersion

extern "C" {
#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2007+2) //The "#els
    static int CHK_OPT(MC_CIRPP)(void *Opt, void *Mod)
    {
        return NONACTIVE;
    }
    int CALC(MC_CIRPP)(void *Opt, void *Mod, PricingMethod *Met)
    {
        return AVAILABLE_IN_FULL_PREMIA;
    }
#else
    int CALC(MC_CIRPP)(void *Opt, void *Mod, PricingMethod *Met)
    {
        TYPEOPT *ptOpt = (TYPEOPT *)Opt;
        TYPEMOD *ptMod = (TYPEMOD *)Mod;

```

```

return mc_cirpp(
    ptMod->flat_flag.Val.V_INT,
    ptMod->T.Val.V_DATE,
    ptMod->InitialYieldsR.Val.V_PDOUBLE,
    ptMod->aR.Val.V_DOUBLE,
    ptMod->bR.Val.V_DOUBLE,
    ptMod->SigmaR.Val.V_PDOUBLE,
    ptMod->InitialYieldsI.Val.V_PDOUBLE,
    ptMod->aI.Val.V_DOUBLE,
    ptMod->bI.Val.V_DOUBLE,
    ptMod->SigmaI.Val.V_PDOUBLE,
    ptMod->Rho.Val.V_PDOUBLE,
    ptOpt->Maturity.Val.V_DATE,
    ptOpt->NbPayement.Val.V_PINT,
    ptOpt->Recovery.Val.V_PDOUBLE,
    Met->Par[0].Val.V_INT,
    Met->Par[1].Val.V_DOUBLE,
    Met->Par[2].Val.V_DOUBLE,
    Met->Par[3].Val.V_ENUM.value,
    &(Met->Res[0].Val.V_DOUBLE),
    &(Met->Res[1].Val.V_DOUBLE));
}

static int CHK_OPT(MC_CIRPP)(void *Opt, void *Mod)
{
    /* temporairement inactive à cause d'un bug (jl) */
    return NONACTIVE;

    return strcmp(((Option *)Opt)->Name, "CreditDefaultSwap");
}

#endif //PremiaCurrentVersion
static int MET(Init)(PricingMethod *Met, Option *Opt)
{
    if (Met->init == 0)
    {
        Met->init = 1;

        Met->Par[0].Val.V_INT = 100;
        Met->Par[1].Val.V_DOUBLE = 1e-4;
    }
}

```

```

        Met->Par[2].Val.V_DOUBLE = 1;
        Met->Par[3].Val.V_ENUM.members = &PremiaEnumRNGs;
        Met->Par[3].Val.V_ENUM.value = 0;
    }

    return OK;
}

PricingMethod MET(MC_CIRPP) =
{
    "MC_CIRPP",
    {
        {"Nsim", INT, {100}, ALLOW},
        {"precision", DOUBLE, {0}, ALLOW},
        {"barrier", DOUBLE, {0}, ALLOW},
        {"RandomGenerator", ENUM, {0}, ALLOW },
        {" ", PREMIA_NULLTYPE, {0}, FORBID}
    },
    CALC(MC_CIRPP),
    {
        {"CDS Spread", DOUBLE, {100}, FORBID},
        {"CDS Spread StdDev", DOUBLE, {100}, FORBID},
        {" ", PREMIA_NULLTYPE, {0}, FORBID}
    },
    CHK_OPT(MC_CIRPP),
    CHK_ok,
    MET(Init)
} ;
}

```