

## Help

```

#include "cgmy1d_stdg.h"
#include "pnl/pnl_vector.h"
#include "pnl/pnl_fft.h"
#include "math/wienerhopf.h"
#include "pnl/pnl_cdf.h"
#include "pnl/pnl_random.h"
#include "pnl/pnl_specfun.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2012+2) //The "#els
static int CHK_OPT(AP_CGMY_SWING_WIENERHOPF)(void *Opt, void *Mod)
{
    return NONACTIVE;
}

int CALC(AP_CGMY_SWING_WIENERHOPF)(void *Opt, void *Mod, PricingMethod *Met)
{
    return AVAILABLE_IN_FULL_PREMIA;
}
#else

static int SwingWienerHopf(int am, double Spot, NumFunc_1 *p, double T, int Nd,
{
    double Strike;
    double cnu, lp1, lm1, lpnu, lmnu, ptprice1, ptdelta1, mu, qu, om;
    int ifCall;

    //Put Case
    ifCall = 0;

    if (M < 2 || G <= 1 || Y >= 2 || Y == 0)
    {
        printf("Invalid parameters. We must have M>=2, G>1, 0<Y<2\ n");
    }

    Strike = p->Par[0].Val.V_DOUBLE;

    lm1 = -M;
    lp1 = G;

```

```

om = 0.0;

cnu = C * pnl_tgamma(-Y);

lpnu = exp(Y * log(lp1));
lmnu = exp(Y * log(-lm1));

mu = r - divid + cnu * (lpnu - exp(Y * log(lp1 + 1.0))) + cnu * (lmnu - exp(Y

qu = r + (pow(lp1, Y) - pow(lp1 + om, Y)) * cnu + (pow(-lm1, Y) - pow(-lm1 - o

swing(1, mu, qu, om, ifCall, Spot, lm1, lp1,
      Y, Y, cnu, cnu, r, divid, T, h, Strike, del, Nd, er, step, &ptprice1, &p

//Price
*ptprice = ptprice1;

//Delta
*ptdelta = ptdelta1;

return OK;
}

int CALC(AP_CGMY_SWING_WIENERHOPF)(void *Opt, void *Mod, PricingMethod *Met)
{
    TYPEOPT *ptOpt = (TYPEOPT *)Opt;
    TYPEMOD *ptMod = (TYPEMOD *)Mod;
    double r, divid;

    r = log(1. + ptMod->R.Val.V_DOUBLE / 100.);
    divid = log(1. + ptMod->Divid.Val.V_DOUBLE / 100.);

    return SwingWienerHopf(ptOpt->EuOrAm.Val.V_BOOL, ptMod->S0.Val.V_PDOUBLE,
                           ptOpt->PayOff.Val.V_NUMFUNC_1, ptOpt->Maturity.Val.V_DA
}

```

```

static int CHK_OPT(AP_CGMY_SWING_WIENERHOPF)(void *Opt, void *Mod)
{
    Option *ptOpt = (Option *)Opt;
    TYPEOPT *opt = (TYPEOPT *) (ptOpt->TypeOpt);

    if ((opt->EuOrAm).Val.V_BOOL == AMER)
        return OK;

    return WRONG;
}

```

```

#endif //PremiaCurrentVersion
static int MET(Init)(PricingMethod *Met, Option *Opt)
{
    static int first = 1;

    if (first)
    {
        Met->Par[0].Val.V_PDOUBLE = 0.001;
        Met->Par[1].Val.V_PDOUBLE = 1.;
        Met->Par[2].Val.V_INT2 = 100;
        first = 0;
    }

    return OK;
}

```

```

PricingMethod MET(AP_CGMY_SWING_WIENERHOPF) =
{
    "AP_CGMY_SWING_WIENERHOPF",
    { {"Space Discretization Step", DOUBLE, {500}, ALLOW}, {"Scale parameter", DOU
        {" ", PREMIA_NULLTYPE, {0}, FORBID}
    },
    CALC(AP_CGMY_SWING_WIENERHOPF),
    {{"Price", DOUBLE, {100}, FORBID}, {"Delta", DOUBLE, {100}, FORBID}, {" ", PRE
    CHK_OPT(AP_CGMY_SWING_WIENERHOPF),
    CHK_split,
    MET(Init)
};

```

