

## [Help](#)

```
#include <stdlib.h>
#include "
href../../mod/hullwhite1d/hullwhite1d_std/hullwhite1d_std_h_src.pdfhullwhit
#include "
href../../mod/hullwhite1d/hullwhite1d_std/hullwhite1d_includes_h_src.pdfhull

//The "#else" part of the code will be freely available after the (year of creat
#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2007+2)
int CALC(CF_FloorHW1D)(void *Opt, void *Mod, PricingMethod *Met)
{
    return AVAILABLE_IN_FULL_PREMIA;
}
static int CHK_OPT(CF_FloorHW1D)(void *Opt, void *Mod)
{
    return NONACTIVE;
}
#else

/**/* Floor price as a combination of ZC Call option prices
static int cf_floor1d(int flat_flag, double r_t, char *curve, double a, double s
                    double Nominal, double K, double periodicity, double first
{
    double sum, tim, tip, strike_call;
    int i, nb_payement;
    ZCMarketData ZCMarket;

    /* Flag to decide to read or not ZC bond datas in "initialyields.dat" */
    /* If P(0,T) not read then P(0,T)=exp(-r0*T) */
    if (flat_flag == 0)
    {
        ZCMarket.FlatOrMarket = 0;
        ZCMarket.Rate = r_t;
    }

    else
    {
        ZCMarket.FlatOrMarket = 1;
        ZCMarket.filename = curve;
    }
}
```

```

    ReadMarketData(&ZCMarket);

    if (contract_maturity > GET(ZCMarket.tm, ZCMarket.Nvalue - 1))
    {
        printf("\ nError : time bigger than the last time value entered in ini
        exit(EXIT_FAILURE);
    }
}

strike_call = 1. / (1 + periodicity * K);
nb_payment = (int)((contract_maturity - first_payment) / periodicity);

/*Floor=Portfolio of zero-bond Call options*/
sum = 0.;
for (i = 0; i < nb_payment; i++)
{
    tim    = first_payment + (double)i * periodicity;
    tip    = tim + periodicity;

    sum    += cf_hw1d_zbcall(&ZCMarket, a, sigma, tip, tim, strike_call);
}

sum = Nominal * (1. + K * periodicity) * sum;

/*Price*/
*price = sum;

DeleteZCMarketData(&ZCMarket);

return OK;
}

int CALC(CF_FloorHW1D)(void *Opt, void *Mod, PricingMethod *Met)
{
    TYPEOPT *ptOpt = (TYPEOPT *)Opt;
    TYPEMOD *ptMod = (TYPEMOD *)Mod;

    return cf_floor1d(ptMod->flat_flag.Val.V_INT,
                      MOD(GetYield)(ptMod),
                      MOD(GetCurve)(ptMod),

```

```

        ptMod->a.Val.V_DOUBLE,
        ptMod->Sigma.Val.V_PDOUBLE,
        ptOpt->Nominal.Val.V_PDOUBLE,
        ptOpt->FixedRate.Val.V_PDOUBLE,
        ptOpt->ResetPeriod.Val.V_DATE,
        ptOpt->FirstResetDate.Val.V_DATE - ptMod->T.Val.V_DATE,
        ptOpt->BMaturity.Val.V_DATE - ptMod->T.Val.V_DATE,
        &(Met->Res[0].Val.V_DOUBLE));
    }
static int CHK_OPT(CF_FloorHW1D)(void *Opt, void *Mod)
{
    return strcmp(((Option *)Opt)->Name, "Floor");
}
#endif //PremiaCurrentVersion

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

    return OK;
}

PricingMethod MET(CF_FloorHW1D) =
{
    "CF_HullWhite1d_Floor",
    {" ", PREMIA_NULLTYPE, {0}, FORBID}},
    CALC(CF_FloorHW1D),
    {"Price", DOUBLE, {100}, FORBID}, {" ", PREMIA_NULLTYPE, {0}, FORBID}},
    CHK_OPT(CF_FloorHW1D),
    CHK_ok,
    MET(Init)
} ;

```