

[Help](#)

```
#include "
href../../../../mod/hullwhite1dgeneralized/hullwhite1dgeneralized_std/hullwhite1dg

#include "
href../../../../common/math/read_market_zc/InitialYieldCurve_h_src.pdfmath/read_mar
#include "
href../../../../mod/hullwhite1dgeneralized/hullwhite1dgeneralized_std/hullwhite1dg

//The "#else" part of the code will be freely available after the (year of creat
#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2010+2)
int CALC(CF_FloorHW1dG)(void *Opt, void *Mod, PricingMethod *Met)
{
    return AVAILABLE_IN_FULL_PREMIA;
}
static int CHK_OPT(CF_FloorHW1dG)(void *Opt, void *Mod)
{
    return NONACTIVE;
}
#else

/*/* Floor price as a combination of ZC Put option prices
static int cf_floor1d(int flat_flag, double r_t, char *curve, int CapletCurve, d
{
    double sum, sigma_avg, T, S;
    int i, nb_payement;

    ModelHW1dG HW1dG_Parameters;
    ZCMarketData ZCMarket;
    MktATMCapletVolData MktATMCapletVol;

    /* 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);
    }
}

ReadCapletMarketData(&MktATMCapletVol, CapletCurve);

hwldg_calibrate_volatility(&HWldG_Parameters, &ZCMarket, &MktATMCapletVol, a);

/*Floor = sum of floorlets*/
nb_payement = (int)((contract_maturity - floor_reset_date) / periodicity);

sum = 0.;
for (i = 0; i < nb_payement; i++)
{
    T = floor_reset_date + (double)i * periodicity;
    S = T + periodicity;
    sigma_avg = hwldg_fwd_zc_average_vol(&HWldG_Parameters, T, S);

    sum += hwldg_floorlet_price(&ZCMarket, sigma_avg, floor_strike, periodici
}

sum *= Nominal;

/*Price*/
*price = sum;

DeleteZCMarketData(&ZCMarket);
DeleteMktATMCapletVolData(&MktATMCapletVol);
DeletModelHWldG(&HWldG_Parameters);

return OK;
}

```

```

int CALC(CF_FloorHW1dG)(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->CapletCurve.Val.V_ENUM.value,
                      ptMod->a.Val.V_DOUBLE,
                      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_FloorHW1dG)(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;
        Met->HelpFilenameHint = "cf_hullwhite1dgeneralized_floor";
    }

    return OK;
}

PricingMethod MET(CF_FloorHW1dG) =
{
    "CF_HullWhite1dG_Floor",
    {{ " ", PREMIA_NULLTYPE, {0}, FORBID}},
    CALC(CF_FloorHW1dG),

```

```
{{"Price", DOUBLE, {100}, FORBID}, {" ", PREMIA_NULLTYPE, {0}, FORBID}},  
CHK_OPT(CF_FloorHW1dG),  
CHK_ok,  
MET(Init)  
} ;
```