

[Help](#)

```
#include "sg1d_std.h"
#include "math/read_market_zc/InitialYieldCurve.h"
#include "QuadraticModel.h"

//The "#else" part of the code will be freely available after the (year of creat
#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2007+2)
int CALC(CF_CapSG1D)(void *Opt, void *Mod, PricingMethod *Met)
{
    return AVAILABLE_IN_FULL_PREMIA;
}
static int CHK_OPT(CF_CapSG1D)(void *Opt, void *Mod)
{
    return NONACTIVE;
}
#else

/// Cap price as a combination of ZC Put option prices
static int cf_cap_sg1d(int flat_flag, double r_t, char *curve, double beta, doub
{
    double sum, T, S, strike_put;
    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_put = 1. / (1 + periodicity * K);
nb_payement = (int)((contract_maturity - first_payement) / periodicity);

/*Cap=Portfolio of zero-bond Put options*/
sum = 0.;
for (i = 0; i < nb_payement; i++)
{
    T = first_payement + (double)i * periodicity;
    S = T + periodicity;

    sum += zb_put_quad1d(&ZCMarket, beta, sigma, T, S, strike_put);
}

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

/*Price*/
*price = sum;

DeleteZCMarketData(&ZCMarket);

return OK;
}

int CALC(CF_CapSG1D)(void *Opt, void *Mod, PricingMethod *Met)
{
    TYPEOPT *ptOpt = (TYPEOPT *)Opt;
    TYPEMOD *ptMod = (TYPEMOD *)Mod;

    return cf_cap_sg1d(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_CapSG1D)(void *Opt, void *Mod)
{
    return strcmp(((Option *)Opt)->Name, "Cap");
}
#endif //PremiaCurrentVersion

static int MET(Init)(PricingMethod *Met, Option *Opt)
{
    if (Met->init == 0)
    {
        Met->init = 1;
        Met->HelpFilenameHint = "cf_quadratic1d_cap";
    }

    return OK;
}

PricingMethod MET(CF_CapSG1D) =
{
    "CF_SquareGaussian1d_Cap",
    {{ " ", PREMIA_NULLTYPE, {0}, FORBID}},
    CALC(CF_CapSG1D),
    {{ "Price", DOUBLE, {100}, FORBID}, { " ", PREMIA_NULLTYPE, {0}, FORBID}},
    CHK_OPT(CF_CapSG1D),
    CHK_ok,
    MET(Init)
} ;

```