

[Help](#)

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

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2007+2) //The "#els
static int CHK_OPT(CF_ZCPutBondEuroSG1D)(void *Opt, void *Mod)
{
    return NONACTIVE;
}
int CALC(CF_ZCPutBondEuroSG1D)(void *Opt, void *Mod, PricingMethod *Met)
{
    return AVAILABLE_IN_FULL_PREMIA;
}
#else

double zb_put_quad1d(ZCMarketData *ZCMarket, double beta, double sigma, double T)
{
    double r0, x0, POT, POS;
    double put_price, p1, p2;

    Data data1, data2;
    Omega om;
    Chn chn;

    r0 = 0.0;
    x0 = 0.0;
    p1 = 0.0;
    p2 = 0.0;

    initial_short_rate(ZCMarket, &r0, &x0);

    // coefficients of P(0,T)
    bond_coeffs(ZCMarket, &data1, T, beta, sigma, x0);

    // coefficients of P(0,S)
    bond_coeffs(ZCMarket, &data2, S, beta, sigma, x0);

    // omega distribution of P(S,T)
```

```

transport(&om, data1, data2, beta, sigma, x0);

// transforms the omega distribution of P(s,T) into chi2 corresponding form */
om2chn(om, &chn);

// Price of the Call option
pnl_cdfbchi2n(-chn.alpha - log(strike), 1, chn.lambda / (1 + 2 * chn.beta), ch
pnl_cdfbchi2n(-chn.alpha - log(strike), 1, chn.lambda, chn.beta, &p2);

// Put Price
POT = BondPrice(T, ZCMarket);
POS = BondPrice(S, ZCMarket);
put_price = POS * (p1 - 1) - POT * strike * (p2 - 1);

if (put_price < 0) put_price = 0.;

return put_price;
}

static int zbc_quad1d(double flat_flag, double beta, double sigma, double r0, ch
{
    double strike;
    ZCMarketData ZCMarket;

    if (flat_flag == 0)
    {
        ZCMarket.FlatOrMarket = 0;
        ZCMarket.Rate = r0;
    }

    else
    {
        ZCMarket.FlatOrMarket = 1;
        ZCMarket.filename = curve;
        ReadMarketData(&ZCMarket);

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

```

```

    }
}

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

*price = zb_put_quad1d(&ZCMarket, beta, sigma, T, S, strike);

DeleteZCMarketData(&ZCMarket);

return OK;
}

int CALC(CF_ZCPutBondEuroSG1D)(void *Opt, void *Mod, PricingMethod *Met)
{
    TYPEOPT *ptOpt = (TYPEOPT *)Opt;
    TYPEMOD *ptMod = (TYPEMOD *)Mod;

    return zbc_quad1d(ptMod->flat_flag.Val.V_INT,
                      ptMod->a.Val.V_DOUBLE,
                      ptMod->Sigma.Val.V_PDOUBLE,
                      MOD(GetYield)(ptMod),
                      MOD(GetCurve)(ptMod),
                      ptOpt->BMaturity.Val.V_DATE - ptMod->T.Val.V_DATE,
                      ptOpt->OMaturity.Val.V_DATE - ptMod->T.Val.V_DATE,
                      ptOpt->PayOff.Val.V_NUMFUNC_1,
                      &(Met->Res[0].Val.V_DOUBLE));
}

static int CHK_OPT(CF_ZCPutBondEuroSG1D)(void *Opt, void *Mod)
{
    return strcmp(((Option *)Opt)->Name, "ZeroCouponPutBondEuro");
}

#endif //PremiaCurrentVersion
static int MET(Init)(PricingMethod *Met, Option *Opt)
{
    if (Met->init == 0)
    {
        Met->init = 1;
        Met->HelpFilenameHint = "cf_quadratic1d_zbputeuro"

```

```

    }
    ;

    }

    return OK;
}

PricingMethod MET(CF_ZCPutBondEuroSG1D) =
{
    "CF_SquareGaussian1d_ZBPutEuro",
    {{" ", PREMIA_NULLTYPE, {0}, FORBID}},
    CALC(CF_ZCPutBondEuroSG1D),
    {{"Price", DOUBLE, {100}, FORBID}, {" ", PREMIA_NULLTYPE, {0}, FORBID}},
    CHK_OPT(CF_ZCPutBondEuroSG1D),
    CHK_ok,
    MET(Init)
} ;
```